DOCUMENT RESUME

ED 225 546 IR 010 551

AUTHOR	Earl, Lorna L.
TITLE	S.P.S.S. User's Manual #1-#4. Basic Program Contruction in S.P.S.S.; S.P.S.S. Non-Procedural Statements and Procedural Commands; System Control Language and S.P.S.S.; Quick File Equate Statement Reference.
INSTITUTION	State Univ. of New York, Plattsburgh. Coll. at Plattsburgh.
PUB DATE	82
NOTE	147p.
PUB TYPE	Guides - Non-Classroom Use (055)
EDRS PRICE	MF01/PC06 Plus Postage.
DESCRIPTORS	*Computer Oriented Programs; *Computer Programs; *Data Analysis; Guides; *Programing; Programing Languages; *Research Methodology
IDENTIFIERS	*Statistical Package for the Social Sciences

ABSTRACT

This series of manuals describing and illustrating the Statistical Package for the Social Sciences (SPSS) was planned as a self-teaching instrument, beginning with the basics and progressing to an advanced level. Information on what the searcher must know to define the data and write a program for preliminary analysis is contained in manual 1, which describes general rules of statement preparation, SPSS conventions, rules of precedence, basic program construction for temporary runs, and permanent file creation and access. Manual 2 contains the variety of data modification and transformational commands, as well as procedural task request commands. A chart outlines general SPSS functions and the control word used to activate the request, and the options that can be added to a basic program to conduct a specialized analysis are listed. Manual 3 explains how to use SPSS on the Burroughs B6800 computing system and describes the necessary file equate statements to alert the system to data location, status of the program, output destination, and the kind of output desired. Instructions are provided for using the terminals, programming on cards, and submitting a job batch. The final manual presents a quick reference chart of file equate statements with examples illustrating how to use the chart. (Author/LMM)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED225546

IR010051

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)

☐ This document has been reproduced as received from the person or organization originating it.

☒ Minor changes have been made to improve reproduction quality.

- Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

S.P.S.S. USER'S MANUAL # 1

Basic Program Construction in S.P.S.S.

Prepared by

Lorna L. Earl
SUNY at Plattsburgh
Spring 1982

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Lorna L. Earl

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

COVER SHEET FOR THE S.P.S.S. MANUAL SERIES

The Statistical Package for the Social Sciences (S.P.S.S.), although designed to be simple and easily understood by "non-computer scientists," tends to intimidate the beginning researcher with the variety of options it offers to conduct data analysis. The advantages of its scope and flexibility unfortunately create the disadvantage of confusing the user as to what does and does not apply to his or her particular research needs. In addition, S.P.S.S. must be employed within the confines of a unique computer system with a special control language that must be mastered prior to running any program.

In an attempt to clarify S.P.S.S. and minimize intimidation, a series of four manuals has been prepared. The voluminous amount of information presented in the complete text explaining S.P.S.S. has been reduced to a set of basic instructions for preparing an S.P.S.S. program and modifying that program to accomplish the desired analysis.

The series is arranged in four manuals that increase in the level of user sophistication. Each document can exist as a separate entity to be used alone or they can be taken together as a complete four-part reference. Manual #1 is written for the beginning user: general rules of statement preparation, conventions used in S.P.S.S., rules of precedence, basic program construction for temporary runs, and permanent file creation and access are addressed. Everything the researcher must know to define the data and write a program for preliminary analysis can be found in Manual #1.

The variety of data modification and transformation commands as well as procedural task request commands can be found in Manual #2. A chart is provided outlining general functions that S.P.S.S. performs and the control word used to activate the request. The control words are arranged in alphabetical order with the non-procedural commands first (e.g. recoding of variable values, creating new variables) and the procedural commands following the last non-procedural command (e.g. descriptive statistical sub-programs, statistical analysis sub-programs). For each control word, the function, status (or applicable variable type), syntax, special instructions for use and where to find more detailed information is presented. Where the first manual illustrates the preparation of a basic program, the second manual contains a listing of all of the options one can add to that program to conduct specialized analysis.

Manual #3 in the series explains how to use S.P.S.S. on the Burrough's B6800 computing system. The necessary system control statements (file equate statements) to alert the system as to the location of the data, status of the program, destination of the output and the kind of output desired are presented in this manual. In addition, instruction is provided for use of the terminals available as well as programming on cards and submitting a job "batch." Everything a researcher needs to know about running S.P.S.S. on this computer can be found in the third manual of the series.

Finally, the last manual abbreviates the file equate statements presented in Manual #3 into a chart thereby providing a quick and easy reference for the advanced user of S.P.S.S. Examples are provided to illustrate how to use the chart.

This series of manuals describing and illustrating S.P.S.S. is intended to be a self-teaching instrument. It begins with the basics and progresses to an advanced level of use. Throughout these manuals are citations of references for complete information on the command being discussed. The major reference for S.P.S.S. is The Statistical Package for the Social Sciences Second Edition by Norman H. Nie, et. al., New York: McGraw-Hill Book Company, 1975. Several other manuals have been produced by the same company: S.P.S.S. Primer by William R. Klecka, et. al, 1975; S.P.S.S. Update 7-9 by C. Hadlai Hull and Norman H. Nie, 1981; and S.P.S.S. Introductory Guide: Basic Statistics and Operations by Marija J. Norusis, 1982. In general, the large manual should be used only to verify syntax or to find out more information on certain commands--it contains far too much and varied information to be used as a reading and teaching text. The Primer provides a simplistic approach to introducing the basics of S.P.S.S. and is a good place to start for the beginner. The Introductory Guide contains an excellent explanation of the use and applicability of a variety of statistical sub-programs available in S.P.S.S. New sub-programs and changes in existing commands are presented in the Update manual. Many of the new facilities and options now available have been included in this series and citations in the Update have been made where applicable.

As a final note from someone who learned this language through trial and error and error and error, good luck and happy programming!

GLOSSARY OF TERMS

Several terms will be used throughout this series with which the user of S.P.S.S. may or may not be familiar. To avoid any confusion about this specialized terminology, the following glossary was prepared.

ALPHANUMERIC: a variable with values that are either letters or special characters.

CASE: the basic unit of analysis in S.P.S.S. - the unit about which a series of information has been gathered (see Example 1 at end).

CHARACTERS: any symbol found in the data - either numbers, letters or special characters such as a '?' or '!'.

COMMAND: a line in a program that activates a statistical procedure. Statistical analysis is done by requesting the appropriate command (e.g. CROSSTABS, REGRESSION).

DATA FILE: the "place" raw data records reside (usually on tape or disk) (see Example 2 at end).

DEFAULT: the value or keyword that will be used if a value or keyword is omitted from the syntax.

FIELD: the column width (number of columns a value takes up) of a variable as it exists on the data record.

FILE EQUATE STATEMENT: the part of the system control language segment of a program that informs the system where, under what name and how to perform specialized functions needed for the program. Each possible function has an associated file number in the form of FILE FILE#.

INPUT: anything entered into the computer for analysis - the program and the data are both forms of input.

JOB: the term used when referring to a run of S.P.S.S.

JOB FILE: the term used in reference to the "place" a program exists on disk (see Example 2 at end).

KEYWORD: a special word used in S.P.S.S. that means something specific to the system - must be spelled and placed precisely as illustrated.

NUMERIC: a variable with values that are numbers (whole or integer).

OUTPUT: anything that the computer returns as a result of an S.P.S.S. program - the results on computer paper, a saved system file, new data that was written are forms of output.

RAW DATA: pieces of coded information from social science research (e.g. a survey) that are in a form that the computer can read and analyze.

RECORD: refers to one line of data. A record can be a data card or a line in a disk data file. Record is used instead of "card" or "card image."

STATEMENT: a line in a program that serves a purpose other than to activate a statistical procedure. Data definition, program documentation, getting into the computing system and running the program are all accomplished by using the correct statements. All non-procedural requests are referred to as statements (as opposed to "commands" which are procedural requests).

SUBPROGRAM: a specific set of programming statements that comprise one statistical procedure. Each statistical request (e.g. FREQUENCIES) is referred to as a subprogram within the general S.P.S.S. package.

SYNTAX: the precise manner that a statement or command specification must be entered in a program.

SYSTEM FILE: the permanent (saved) version of the information created during a run of S.P.S.S. (described in detail in Manual 1) - used when referring to any permanent or saved S.P.S.S. file (see Example 2 at end).

VALUE: one of a possible range of numbers or alphanumeric characters that a variable has assigned to it for a specific case (see Example 1 at end).

VARIABLE: an observable entity which can take on more than one value or characteristic (see Example 1 at end).

EXAMPLE 1: Differenciating a "case," "variable" and "value"

A data set has 100 cases (people who responded to a survey). For each case, several pieces of information or variables exist (demographics of the respondent, each question that was asked, an identification number). The actual response to each question on the survey is the value of the variable. For example, the first case is a 21 year old female and she answered "yes" to questions 1, 2 and 3. Twenty-one is the value for the variable "age," for that case; female is the value for the variable "sex" for that case; and "yes" is the value for the three variables "question 1" "question 2" and "question 3" for that case.

EXAMPLE 2: Differenciating a "Data file," "Job file" and "system file." The raw data records for the information from the above 100 surveys exist in a data file (usually on disk or tape). The program that defines this data in a way that the computer can read and analyze resides in the Job file on disk (analogous to a card deck on which a program is keypunched). The job file has a statement in it that tells the system to find the data file and read the data to perform the analysis. If the defining information is to be made permanent through S.P.S.S., this permanent, internal machine-coded information is stored in a system file.

Table of Contents

Introduction to the Manual.. 3

General Introduction to S.P.S.S. Programming........................ 4

Preparing the Data for Input.. 4

Naming Variables.. 5

Control Statement Preparation....................................... 6

Conventions for Lists... 6

Other General Rules... 8

About S.P.S.S. Files.. 8

Necessary Commands to Build an S.P.S.S. File........................ 10

- RUN NAME.. 10
- FILE NAME... 11
- DATA LIST... 11
- VARIABLE LIST and INPUT FORMAT.................................... 14
- INPUT MEDIUM.. 17
- N OF CASES (and END INPUT DATA)................................... 17
- MISSING VALUES.. 18
- VAR LABELS.. 20
- VALUE LABELS.. 21
- READ INPUT DATA... 22
- FINISH.. 22

Necessary Commands to Build a Permanent S.P.S.S. File............... 23

- SAVE FILE... 23

Necessary Commands to Access a Permanent File....................... 24

- GET FILE.. 24

Modifying a File and Saving the New File............................ 24

Special Consideration for Alphanumeric Data and Integer Data......... 25

Dealing with Subfile Structures..................................... 26

Dealing with Archive Files.. 27

Conclusion.. 30

Appendices

Appendix 1
Summary of S.P.S.S. Control Statements:
Precedence of Appearance in a Program 31
Appendix 2A
General Program Construction: Temporary File 33
Appendix 2B
General Program Construction: Permanent File Construction 34
Appendix 2C
General Program Construction: Accessing a Permanent File 35
Appendix 3
Reference List 36

INTRODUCTION

This manual is written for the person interested in conducting statistical analysis on social science data. Previous use of S.P.S.S. (the Statistical Package for the Social Sciences) is not presupposed, but a familiarity with the issues involved in conducting research is prerequisite to effective use of this reference. The purpose and usefulness of this particular computer package in research, the methodology involved in research, and appropriate statistical analysis for varying kinds of data are the necessary background information for use of the material addressed here. For a presentation of these issues, see Statistical Package for the Social Sciences, Second Edition by Norman H. Nie, et. al. Chapter 1 and 2 (complete reference in Appendix #3). Once the user is familiar with the information in those chapters, this manual can be employed to help the beginner construct a basic S.P.S.S. program.

This first manual, then, presents the necessary S.P.S.S. control statements for the construction of a simple program. Program documentation (RUN NAME, FILE NAME) data definition (DATA LIST, VARIABLE LIST and INPUT FORMAT, INPUT MEDIUM, N OF CASES, MISSING VALUES, VAR LABELS, VALUE LABELS) creation and access of permanent files (SAVE FILE, GET FILE) and run statements (READ INPUT DATA, FINISH) are the components of an S.P.S.S. program that will be described and illustrated. In addition, general rules for organizing the raw data, statement preparation, conventions for variable and value lists and other S.P.S.S. guidelines will be presented in this manual to prepare the beginning programmer for correct use of S.P.S.S.

GENERAL INTRODUCTION TO S.P.S.S. PROGRAMMING

The S.P.S.S. system is an integrated package of computer programs that allow a variety of data manipulations and statistical analysis on social science data. An S.P.S.S. program is the vehicle through which the researcher conducts analysis. Any program contains the following groups of statements:

1. Job or System control statements
2. Data-definition statements
3. Task-definition commands
4. Data records

The system control statements are the statements that allow access to the computer that are unique to each system and are presented in Manual #3 of this series. Data-definition statements identify the name, location, size and type of each variable (addressed in this manual). Manual #2 presents task-definition commands which specify the statistical procedures desired as well as the range of data-modification statements available in S.P.S.S. Finally, every program must have a set of data associated with it upon which the analysis is conducted. This social science data must be coded in a way such that the computer can "read" the responses.

PREPARING THE DATA FOR INPUT

There are several guidelines the researcher should follow for coding of the raw data into a form that the computer can efficiently "read" and manipulate.

1. It is recommended that each variable have numeric values representing each possible response (example: a "yes" response could equal '1' and a "no" could be coded as a '0').
2. Numeric values cannot exceed 9 columns and alphanumeric variables cannot be larger than 4 columns.
3. There are 2 options for organizing the data on the actual records:
   a. Fixed-column format (recommended) - the value of every variable for every case is located on the same record in the same column(s). Each case has the same number of variables and the same number of data records for each case.

   b. Free-field format - the location of the variables for each case is irrelevant (the column numbers or record numbers need not correspond among cases), the sequence of variables on each record must be the same. Each case has the same number of variables but not necessarily the same number of records per case.

4. When assigning codes to responses, each value must be unique.
5. There are special conventions for entering the data:
   a. Blanks between variables are not necessary but can be inserted to make the data record easier to read.
   b. For decimal numbers, the decimal point can be punched in its place on the data record or may be omitted (to be defined later in the program).
   c. Alphanumeric characters are entered as they exist but, are always enclosed in single quotation marks when referenced in the program.
   d. In a fixed-format, always allow a large enough field (column width) for each variable (e.g. number of children should be alloted more than one column because it is possible to have more than 9 children).
   e. When entering data with alloted fields larger than the actual value, right justify the value. Optional zeros can be placed to the left of the value to fill in the space or blanks can be left.
   f. Blanks are read by the computer as zeros.
   g. Values may never be split between records and a single value cannot have imbedded blanks (since a blank is read as a zero an imbedded blank would alter the value read by the computer).
   h. If a variable has a negative value, the negative sign (dash) must be included on the data record and calculated in the alloted field width. If no sign is supplied, the value is assumed positive.
   i. To help keep the data in order (especially if there are several records per case), it is wise to allow several columns for an identification number on at least the first record of a case and record numbers on each record in the case.

NAMING VARIABLES

Every piece of information must be assigned an S.P.S.S. variable name which will be used throughout the program. These are the rules for naming variables:

1. The name must be unique.
2. It can range from 1 to 8 characters in length.
3. The first character must be an alphabetic letter.

4. Otherwise, the name can consist of numbers or letters or a combination of both.
5. There can be no special characters and no imbedded blanks in a name.
6. Names can never be split between records.

CONTROL STATEMENT PREPARATION

Each control statement is comprised of two fields: the control field (always starts in column 1 and is never longer than 15 characters) and the specification field (beginning in column 16 and continuing through column 80 or onto the next physical record(s) columns 16-80). The control words in the first 15 columns must be spelled and spaced exactly as they are illustrated. Anything appearing in the specification field (col. 16-80) must be separated by delimiters (either "common" or "special"). There are two "common" delimiters:

- a blank
- a comma

Either or both must be used between variable names, values, keywords and/or labels in the specification field. There are four "special" delimiters:

- left parenthesis [(] - must be used in conjunction - usually used for values
- right parenthesis [)]
- slash [/] - used to separate groups of variable names or to signify another set of specifications on the same command
- equal sign [=] - used to separate a created variable or keyword and its specifications

CONVENTIONS FOR LISTS

S.P.S.S. provides the user with several shorthand mechanisms for referencing variables and/or their values when listing them. To use these conventions, the variables and/or values must be contiguous (either in the data defining statement or in the saved system file). The basic conventions for shortening lists are:

1. For variables being referenced in a data defining statement and anytime after:

```
alphaxxx TO alphayyy
```

where "alpha" is the same string of letters and "xxx" is a number and "yyy" is a larger number.

```
Examples:
          VAR01 TO VAR50
          ITEM12 TO ITEM26
          NUMBER1 TO NUMBER25
```

In these examples, the variable names between the referenced variables are supplied by the system implying that each next variable has the same letter and number sequence plus one. Although this significantly shortens notation, one loses the advantage of naming variables so they can be easily recognized.

2. Regardless of variable names, the "TO convention" can be used anytime after the data definition statement for variables that are contiguously defined.

```
Example:  These variables have been defined in this order on a data-
          definition statement:

             VAR1 TO VAR23, AGE, SEX, EDUC, ITEM1 TO ITEM6, MARITAL

          They can all be referenced later by this list:

             VAR1 TO MARITAL
```

3. For <u>values</u> being referenced, the keywords THRU, LOWEST (LO) and HIGHEST (HI) can be used to shorten notation.

```
Examples:
          (3 THRU 98)
          (2, 5, 8 THRU HIGHEST)
          (LOWEST THRU 50)
```

OTHER GENERAL RULES

1. Every variable referenced in a program must be previously defined (name, location, size, type) in a data-definition statement or in a command that created it.
2. Although most program statements can appear in any order, some commands take precedence over others and must be organized accordingly. Appendix 1 presents a chart of the precedence, status (required, conditional, or optional) and purpose of the various statements found in S.P.S.S. Appendices 2A, 2B, and 2C are charts illustrating the sequences of statements for basic program construction of temporary files, creation of permanent files and the access of a saved file respectively.

   General rules of precedence are as follows:

   a. One procedural command must appear before the statement that activates the reading of the raw data. Other procedural requests are placed after the data records.
   b. S.P.S.S. allows for temporary changes in variables (denoted by an asterisk in column 1 of the control field; e.g. *RECODE will change the values of the specified variable only for the procedure that directly follows it). These temporary statements are placed before the procedure to which they apply and after all permanent data definition and modification statements.
   c. All permanent data modifications and transformations must appear before the first procedural request.

ABOUT S.P.S.S. FILES

Data is manipulated through S.P.S.S. in groupings of information called files. Every time a program is run, a file is created in the internal mechanism of the computer system. The user can chose to retain this grouping of information permanently or leave the file temporary (the information will be "lost" at the end of the run). Regardless of the status of the file for a run of S.P.S.S., the file must contain the following information:

a. a file name (the default name for the file will be 'NONAME');

b. a variable list defining the names, location, size and types of the variables to be included;

c. the subfile structure (if any);

d. the number of data cases in the file (or in each subfile); and

e. the input medium on which the raw data resides.

A file may optionally contain the following:

a. the values that represent missing responses in the data;

b. variable labels;

c. value labels for the variables; and

d. variable printing formats.

The necessary information is entered by the use of data-definition statements and, together with the raw data records, constitute a raw data input file (if temporary) or an S.P.S.S. system file (if made permanent). Permanent files allow a great deal of user flexibility in manipulating sets and sub-sets of data for more efficient use of the computer system. An S.P.S.S. file (temporary or permanent) can handle up to 500 original and/or created variables. If the user wishes to manipulate more than 500 variables, an archive file must be created to handle analyses of up to 5000 variables.

NECESSARY COMMANDS TO BUILD A TEMPORARY S.P.S.S. FILE

This sample program contains both required and optional statements to build a temporary S.P.S.S. file. Following the example, each statement will be fully explained. The order in which each statement appears should be noted because they follow the correct order of precedence.

```
1              16
RUN NAME       THIS IS AN EXAMPLE S.P.S.S. PROGRAM
FILE NAME      TEMPFILE  THIS IS ONLY A TEMPORARY FILE
DATA LIST      FIXED(3)/1 IDNUM 1-3, VAR1 TO VAR6  4-9,
               DOG 11-12, CAT 13/3 CITY 2-4, COUNTRY 5-7, SEX 9, EDUC 10-11,
               INCOME 13-18
INPUT MEDIUM   CARD
N OF CASES     100
MISSING VALUES VAR1 TO VAR6, CAT, SEX (9)/DOG, EDUC (99)/CITY, COUNTRY (888,999)/
               INCOME (999999)
VAR LABELS     VAR3, DO YOU OWN A PET OTHER THAN DOG OR CAT?/DOG, BREED OF DOG
               OWNED/
               CAT, BREED OF CAT OWNED/
               CITY, CITY OF RESIDENCE
VALUE LABELS   VAR1 TO VAR6 (0) NO (1) YES/
               SEX (0) FEMALE (1) MALE
FREQUENCIES    GENERAL= VAR1 TO EDUC
READ INPUT DATA


data records


CONDESCRIPTIVE INCOME
FINISH
```

<u>RUN NAME</u>:

<u>Function</u> - identifies the current computer run by printing the specified label on the top of each page of output

<u>Status</u> - optional

<u>Syntax</u> -

```
1              16
RUN NAME       any label
```

Special Instructions -

1. The label can be from one to sixty-four (64) characters long.
2. Any character can be used in the label (letters, numbers, special symbols)

Further Information - Chapter 6 (p. 72)*

FILE NAME:

Function - identifies the S.P.S.S file created by the run by assigning it a unique name

Status - optional for a temporary run (file is not to be saved)

Syntax -

```
1              16
FILE NAME      name  optional label
```

Special Instructions -

1. The name must follow the rules for naming an S.P.S.S. variable.
2. The optional label serves to document the purpose and/or content of the file and can be from one to sixty-four (64) characters long.
3. If omitted, the file created by the run will have the default name "NONAME."

Further Information - Chapter 4 (p. 36)

DATA LIST:

Function - defines the data by providing a complete map of the content and structure of raw data input

Status - required (the alternative is a VARIABLE LIST in conjunction with INPUT FORMAT - one or the other must be used - See page 16)

Syntax -

```
1              16
DATA LIST      FIXED (3)/# var. col.#(s), var. col.#(s),..., var. col.#(s)/#
               var. col.#(s),..., var. col.#(s)#...
```

---

*Chapter citation refers to sections of the large S.P.S.S. Manual. Statistical Package for the Social Sciences. (Complete reference in Appendix 3)

Special Instructions -

1. Since data are usually coded in a fixed-column BCD format, the fixed-format data list is illustrated. S.P.S.S. provides for defining data that is stored in binary (internal machine code) format. For detailed information on the BINARY format DATA LIST, see pp. 55-57 in the large S.P.S.S. manual.

2. The DATA LIST contains the following information:

   - the format of the data (fixed or binary);
   - the number of records in each data case;
   - the record number(s) upon which the variables selected for definition can be found;
   - variable names;
   - the column number(s) associated with each variable; and
   - the type of variable (if no special specification is made, the variable is assumed to be a non-decimal numeric character).

3. FIXED (#) indicates that the data is in fixed-format and the number of records per case. There is no limit on the number of data records that comprise a case.

```
Examples:          16
                   FIXED (1)
                      or
                   FIXED (3)
                      or
                   FIXED (20)
```

   All of the above are legitimate statements. The number in parenthesis should always be equal to the number of records needed to hold all of the information on each case.

4. Directly following this number in parenthesis is a slash and a number indicating the record number upon which the first variable to be defined resides. If the variables for the run are on the first record, the syntax would be:

```
16
FIXED(#)/1
```

If it was on fifth record:

```
FIXED(#)/5
```

A new record is designated by a slash-number sequence. For example:

```
16
FIXED(12)/1 ~~~~~~~~~~~ /3 ~~~~~~~~~~ /11 ~~~~~~~~~~
```

This data list defines variables on the first, third and eleventh record of a 12 record per case data set.
Any number listed after a slash in the data list can never be greater than the number in parenthesis.

5. Following the "slash-record number" sequence is a list of variables and their corresponding location (column number (s)) on that record.

```
16
FIXED (5)/2 VAR01 3-5, VAR02 6, SEX 7, AGE 12-13/3 TEST1 16, TEST2 18-19
```

- Single digit variables are designated by the column number in which they reside.
- Multiple digit variables are designated by the column number in which they start, a dash, and the column number in which they end.
- Alphanumeric variables are designated by '(A)' after the column number (s):

```
...., ALPHANUM 12-13 (A), NUMERIC 15-18,...
```

- Numeric variables that have a decimal in their field (either implied or physically on the data record) are designated by the number of places to the right of the decimal point in parenthesis after the column numbers:

```
..., DECIMAL 12-16 (2),...
```

The variable DECIMAL has a field width of 5 columns with two places to the right of the decimal point to be found and used in calculations. If the decimal is physically in the data, this notation is optional because the actual decimal place overrides the one designated in a data list. If, the decimal point does not exist physically in the data, this notation is required.

- Variable - location sequences are separated by a comma and/or a space.
- The last variable to be defined signals the end of the data list - nothing follows the final column number.

6. Not every variable in the data needs to be defined in the data list - only those which will be used in the present program. Remember, any variable used in the program <u>must</u> be defined previously.
7. Within each record, variables can be defined in any order regardless of their sequential position. Records, however, must be cited in the data list in the order they reside in the data (e.g. a variable in the middle of record #3 can be defined before variables in the beginning of the record but record #3 must be cited after #1 and #2 and before any record following it).
8. S.P.S.S. provides several shorthand methods for defining variables on the DATA LIST if they satisfy the following criteria:
   - they reside in adjacent columns on the same data record;
   - they have the same field width; and
   - they are of the same type (all whole numerics, all of the same decimal place or all alphanumerics).

   The shorthand notations are:
   - Variables using the "alphaxxx TO alphayyy" convention may be listed in this way:

   ```
   VAR01 TO VAR79 1-79
   ```

   - A series of adjacent, similar size and type variables may be listed in this notation:

   ```
   SEX, DOG, CAT, WHY, BECAUSE 17-31 (A)
   ```

9. The DATA LIST is one way to define the variables in a data set. S.P.S.S. provides another option for this task. By using VARIABLE LIST in conjunction with INPUT FORMAT, variables will be defined in a fortran-style format with the same information as the data list. The following is the deck set up for the initial example using these alternative data-definition statements:

```
1              16
RUN NAME       THIS IS AN EXAMPLE S.P.S.S. PROGRAM
FILE NAME      TEMPFILE THIS IS ONLY A TEMPORARY FILE
VARIABLE LIST  IDNUM, VAR1 TO VAR6, DOG, CAT, CITY, COUNTRY, SEX, EDUC,
               INCOME
INPUT MEDIUM   CARD
N OF CASES     100
INPUT FORMAT   (F3.0,X,6F1.0,X,F2.0,F1.0//X,F3.0,X,F3.0,X,F1.0,F2.0,F6.0)
MISSING VALUES VAR1 TO VAR6, CAT, SEX(9)/DOG, EDUC(99)/CITY, COUNTRY
               (888,999)/ INCOME (999999)
VAR LABELS     VAR3, DO YOU OWN A PET OTHER THAN DOG OR CAT/
               DOG, BREED OF DOG OWNED/
               CAT, BREED OF CAT OWNED/
               CITY, CITY OF RESIDENCE
VALUE LABELS   VAR1 TO VAR6 (0) NO (1) YES/
               SEX (0) FEMALE (1) MALE
FREQUENCIES    GENERAL= VAR1 TO EDUC
READ INPUT DATA

data records

CONDESCRIPTIVE INCOME
FINISH
```

- The VARIABLE LIST replaces the DATA LIST statement but consists of only the variable names in the order they are to be defined.
- The location and type of each variable is designated in the INPUT FORMAT statement which is located after the N OF CASES. This statement is a Fortran statement enclosed in parenthesis mapping the location, size and type of each variable referenced in the VARIABLE LIST. The notation is as follows:
  - Numeric variables are designated by the letter F, alphanumeric variables are designated by A.
  - The column width as well as decimal places is denoted by a number after the letter. For example:

```
...,F3.0, A4, F2.1, F5.3
```

This string represents the following variables: one variable 3 columns wide with no decimal place, a four column wide alphanumeric variable, a 2 column wide variable with one place to the right of the actual or implied decimal point and a 5 column wide variable with 3 places to the right of the decimal point.

- Spaces (blanks) are denoted by 'X'.
- Separate records are denoted by a slash and whole records are skipped by using multiple slashes.
- Multiple specifications are done by placing the number of times the same sequence should appear (e.g. 32X skips 32 spaces; 5F3.0 signals five variables of 3 column widths are being defined; 8(3F2.0, 3X, 3A) denotes eight sequences of "3 two-column wide variables, 3 blanks and 3 one-column wide alphanumerics").

10. Each option (DATA LIST or VARIABLE LIST/INPUT FORMAT) has certain advantages and disadvantages.

- DATA LIST is one statement that contains all of the needed information and closely associates names and locations to reduce errors. Errors are also reduced when defining only a few variables from a multi-record data set. For example, this data list:

```
1              16
DATA LIST      FIXED (45)/13 NEUROTIC, PARANOID, SCHIZO 19-21/29
               NHOSPIT, NOUTPATN 45-45/33 FAMHIST 7
```

would require these two statements to be prepared:

```
VARIABLE LIST  NEUROTIC, PARANOID, SCHIZO, NHOSPIT, HOUTPATN, FAMHIST
INPUT FORMAT   (////////////18X,3F1.0////////////////44X,2F2.0////6X,
               F1.0////////////)
```

The alternative to a data list is most convenient when the user is very familiar with constructing fortran statements or when there are many variable groups of similar type.

The following data list:

```
DATA LIST      FIXED (5)/1 V1 TO V20 1-80 (2)/2 V21 TO V40 1-8 (2)/3
               V41 TO V60 1-80 (2)/4 V61 TO V80 1-80 (2)/5 V81 TO V100
               1-80 (2)
```

can be compressed to

```
VARIABLE LIST  V1 TO V100
INPUT FORMAT   (20F4.2/20F4.2/20F4.2/20F4.2/20F4.2)
```

INPUT MEDIUM:

Function - informs the system where the raw data for the program resides

Status - required

Syntax -

```
1              16
INPUT MEDIUM   keyword
```

Special Instructions -

1. The available keywords are CARD, TAPE or DISK.
2. If the keyword CARD is used, the data records must be physically included in the program directly following the READ INPUT DATA statement.
3. If TAPE or DISK is used, a file-equate statement must be included in the system control statement segment of the program to direct the computing system where to find the data.

Further Instructions - Chapter 4 (pp. 39-40)

N OF CASES:

Function - informs the system of the number of cases that can be found in the data set

Status - required

Syntax -

```
1              16
N OF CASES     # or UNKNOWN
```

Special Instructions -

1. This statement is only used if a file does not have a subfile structure.
2. If the exact number cases is known, the number is listed starting in column 16.
3. If the exact number of cases is not known, the keyword "UNKNOWN" can be used
    - If the raw data exists on disk or tape, this is the only specification needed - when the data file ends, the system will calculate the number of cases.
    - If the raw data exists on cards, an extra statement (END INPUT DATA) is needed after the last data record to signal the end of the data and to read the next statement as an S.P.S.S. command.

```
1              16
INPUT MEDIUM   CARD
N OF CASES     UNKNOWN


READ INPUT DATA


data records


END INPUT DATA
FINISH
```

4. After the data has been read with N OF CASES as UNKNOWN, the printout will contain the precise number of cases read.

MISSING VALUES:

Function - assigns a unique value to missing information on a variable to allow convenient processing - signals the system that, when this value or set of values is encountered, exclude the variable from statistical analysis but retain the case for further analysis

Status - optional

Syntax -

```
1              16
MISSING VALUES variable list (missing value(s))/variable list (missing value(s))/...
```

Special Instructions -

1. Up to three (3) explicit missing values may be declared for a variable or any list of variables.

```
1              16
MISSING VALUES VAR01 TO VAR05, VAR20, VAR22 (7, 8, 9)/VAR6 (0)/VAR7 TO VAR12
               (0,99)/ VAR 13 TO VAR18 (0, 96 THRU 99)
```

   The above is a valid list of missing values. Notice the last list of missing values has three explicit missing values but 5 implicit values that will all be recognized as missing data.

2. Every variable listed on the MISSING VALUES statement must be previously defined.
3. The MISSING VALUES command remains in operation permanently. If the variables and/or value list is to be altered in any way, a complete MISSING VALUE statement must be re-done and this new statement will act as the permanent command.
4. Listing variables:
   - three or more adjacent variables with the same missing value(s) may be listed with the "TO" convention;
   - otherwise, variables with the same missing value(s) must be listed individually and separated by commas.
5. Listing the values:
   - value(s) are enclosed within parenthesis;
   - values are separated by commas;
   - if at least three consecutive values are to be missing values, the "THRU" convention may be used;
   - if a missing value for an alphanumeric variable is to be declared, the missing value(s) must be enclosed within single quotations. For example: ALPHA ('-', '9') .
6. Variable lists with different missing values are separated by a slash.
7. This command may contain any or all variables in the data set and can be continued physically from one record to the next.
8. If all variables in a data set are to have the same missing values, the keyword "ALL" may be used

```
1              16
MISSING VALUES ALL (8, 9)
```

Further Information - Chapter 4 (pp. 57-59)

VAR LABELS -

Function - associates a label with any or all variables in a data set and prints that identifying or descriptive label beside the variable whenever it is referenced on the printed output

Status - optional

Syntax -

```
1                   16
VAR LABELS          var. name, up to 40 character label     /
                    var. name, label                         /
                    var. name, label                         / ...
                    var. name, label
```

Special Instructions -

1. A single variable name is listed then, separated by a comma (or another common delimiter), a descriptive label is found.
2. If more than one variable is to be labeled, the first label is followed by a slash.
3. A label can be up to 40 characters in length (including blanks) and can contain any character with the exception of a slash, comma, left and right parenthesis.
4. A "var. name, label /" sequence need not start at the beginning of a new line (in column 16), it can directly follow the slash on the same record and physically continue to the next record. The syntax is illustrated as a separate line for each variable label because corrections and changes are easier to effect if the labels are organized in this manner.
5. Labels may be physically continued on the next record and words within the label may be split between records.

```
                                                   80
~~~~~~~~~~/ variable name,  THIS LABEL WILL N
OT FIT/ variable name, ....
```

Further Information - Chapter 4 (pp. 62-63)

VALUE LABELS:

Function - associates a label with any or all values of any or all variables in a data set that is printed beside the value whenever it appears on printed output

Status - optional

Syntax -

```
1              16
VALUE LABELS   var.name or list (1st value) up to 20 character label
               (2nd value) up to 20 character label (3rd value) label.../
               var.name or list (1st value) label.../
               var.name or list (1st value) label... (1st value) label
```

Special Instructions -

1. Either one variable can be listed for a particular set of value labels, or several variables that have the set of value labels may be listed.
2. If other variables are to be included on this command that have different value labels, the lists are separated by a slash.

```
VALUE LABELS   VAR01 TO VAR06 (1)~~~~~(2)~~~~~(3)~~~~~/
               VAR07  (1)~~~~~(2)~~~~~(8)~~~~~(9)~~~~~/
               VAR08, VAR10    (0)~~~~~(1)~~~~~
```

3. The labels can be up to 20 characters (including blanks) and can contain any character with the exception of a slash, comma, and left and right parenthesis.
4. For some procedures (CROSSTABS), only the first 16 characters in the label are printed (8 on one line, and 8 on the line below it).
5. Labels can be continued on one record to the next and words in the label only can be split between cards if ended on column 80.

```
Example -

16                                       80
~~~~~~~ (3)~~~~~~~~~(4) THIS LABEL WI
LL NOT FIT (5)~~~~~~~/
```

6. If every variable in the data set has the same value labels, the keyword "ALL" can be used.

```
Example -

1              16
VALUE LABELS   ALL (1)~~~~~~~~~(2)~~~~~~~~ (3)~~~
```

Further Information - Chapter 4 (pp. 59-61)

The "FREQUENCIES" is the one procedural request that must come before the data is read because it activates the reading of the data. This procedure is explained in Manual #2 in this series. Any one procedural request outlined in Manual #2 could be used to activate the command that alerts the system to read the data and accomplish the request.

READ INPUT DATA:

Function - instructs the system to begin reading the raw data input

Status - required

Syntax -

```
1              16
READ INPUT DATA
```

Special Instructions -

1. This command directly follows the first procedural request and is used whenever raw data input is to be read.
2. If the data resides on cards, the data records must directly follow this command (the command is needed regardless of where the data resides - cards tapes or disk).
3. Nothing resides in the specification field (col. 16 through 80).

Further Information - Chapter 5 (p. 67)

FINISH:

Function - terminates the run of S.P.S.S.

Status - required

Syntax -

```
1              16
FINISH
```

Special Instructions -

1. This is always the last statement in an S.P.S.S. program.
2. There is nothing in the specification field.
3. When encountered, the following termination message is printed:

```
NORMAL END OF JOB
XX CONTROL RECORDS WERE PROCESSED
Y ERRORS WERE ENCOUNTERED
```

Further Information - Chapter 6 (p. 73)

NECESSARY COMMANDS TO BUILD A PERMANENT S.P.S.S. FILE

Only one additional command is necessary in any S.P.S.S. program to permanently retain the information generated from the run for future access. This command is:

```
1              16
SAVE FILE
```

It is placed directly before the FINISH command and instructs the system to store all of the information in the file with the name specified on the now required FILE NAME statement.

The following illustrates the example program discussed in the previous section modified to save the information:

```
1              16
RUN NAME       THIS IS ANOTHER EXAMPLE
FILE NAME      PERMFILE  THIS IS NOW A PERMANENT FILE
DATA LIST      FIXED(3) /1 IDNUM 1-3, VAR1 TO VAR6 4-9, DOG 11-12, CAT 13/3 CITY 2-4,
               COUNTRY 5-7, SEX 9, EDUC 10-11, INCOME 13-18
INPUT MEDIUM   CARD
N OF CASES     100
MISSING VALUES VAR1 TO VAR6, CAT, SEX (9)/DOG, EDUC (99)/CITY, COUNTRY (888,999)/
               INCOME (999999)
VAR LABELS     VAR3, DO YOU OWN A PET OTHER THAN DOG OR CAT?/DOG, BREED OF DOG OWNED/
               CAT, BREED OF CAT OWNED/
               CITY, CITY OF RESIDENCE
VALUE LABELS   VAR1 TO VAR6 (0) NO (1) YES/
               SEX (0) FEMALE (1) MALE
FREQUENCIES    GENERAL= VAR1 TO EDUC
READ INPUT DATA


data records


CONDESCRIPTIVE INCOME
SAVE FILE
FINISH
```

Every variable in the DATA LIST, the number of cases, the missing values, labels and data records will all be stored in a file called PERMFILE.* Any variable changed during the run or created during the run will also be saved. Never again will these variables need to be defined or labeled unless a new system file using different original variables (from the raw data) is desired.

---

*Whenever a file is created and/or accessed, a system control statement must be present to identify the file. See Manual #3.

NECESSARY COMMANDS TO ACCESS A PERMANENT FILE

```
1              16
RUN NAME       THIS IS HOW TO ACCESS A FILE
GET FILE       PERMFILE
COUNT          PET = DOG, CAT (1)
ASSIGN MISSING PET (9)
CROSSTABS      TABLES = PET CY CITY, COUNTRY, SEX
STATISTICS     ALL
FINISH
```

The GET FILE directly follows the RUN NAME statement (if a RUN NAME is not present, the GET FILE is the first command in the program). The GET FILE is then followed by any number of data transformations, variable modification and/or creation statements, and procedural requests. Note that GET FILE replaces the DATA LIST, INPUT MEDIUM, N OF CASES and READ INPUT DATA statements because the information already resides within the file.

MODIFYING A FILE AND SAVING THE NEW FILE

As mentioned above, any number of variable modifications and/or creations of new variables can be done to a saved file. These modifications can also be retained in a new, updated system file of a different name. In this way, the original file is retained while a new file is created and saved - each with its own purpose. The following example illustrates this procedure.

```
1              16
RUN NAME       MODIFYING AN OLD FILE
GET FILE       PERMFILE
FILE NAME      NEWFILE  CREATES PET AND NEWVAR AND RECODES EDUC
COUNT          PET = DOG, CAT (1)
ASSIGN MISSING PET (9)
COMPUTE        NEWVAR=(VAR01+3)/VAR02
ASSIGN MISSING NEWVAR(99)
RECODE         EDUC (0 THRU 10=1) (11 THRU 15=2) (16 THRU HIGHEST=3)
FREQUENCIES    GENERAL = PET, NEWVAR
CROSSTABS      TABLES = PET BY NEWVAR BY SEX
STATISTICS     ALL
SAVE FILE
FINISH
```

The file NEWFILE will contain all of the original information for every variable (with the exception of EDUC which now has only 3 values) plus the new variables created by the run. The file will be saved under the name found on the FILE NAME statement. If the FILE NAME is omitted, an alternative to naming the new file is available on the SAVE FILE command.

```
1              16
RUN NAME
GET FILE       PERMFILE
   I
SAVE FILE      NEWFILE  CREATES PET AND NEWVAR AND RECODES EDUC
```

The new file can be named on the SAVE FILE in the same way as it was on the FILE NAME (with or without the file label). Remember, a new file should be given a new name to avoid confusion between original and modified files.

SPECIAL CONSIDERATIONS FOR DECIMAL DATA AND ALPHANUMERIC DATA

Often-times, data may contain values that have decimal places instead of whole numbers. Although S.P.S.S. processes these variables with the decimal places and calculates them correctly, only the number to the left of the decimal (whole number) is printed on the output (if a value is completely fractional, a zero will be printed). The entire decimal number (or part of the decimal) can be printed on the output with one user specification included in the initial program. Alphanumeric values must also be specified before they are processed correctly.

The PRINT FORMATS command indicates how a variable's values are to be printed on the output. The user can control the number of decimal places to the right of the decimal for numeric values and must indicate the presence of an alphanumeric variable. Although optional for decimal numerics, this statement is required for alphanumeric variables. PRINT FORMATS is placed after data-definition and data-modification statements (because created variables can be included) and before the first procedural request. The syntax is as follows:

```
1              16
PRINT FORMATS  variable list (# or A)/variable list (# or A)/...variable list (# or A)
```

The variable list contains all the variables that have the same print format; other variables with different specifications follow separated by a slash. The '#' in parenthesis is a value indicating the number of places to the right of the decimal that are to be printed for numeric variables. For alphanumerics, an 'A' would appear in parenthesis. The following illustrates the syntax and deck placement of this statement:

```
1              16
DATA LIST      FIXED(12)/3 NUMDEC1 TO NUMDEC10 13-53 (2), ALPHA    77-79 (A)
INPUT MEDIUM   DISK
N OF CASES     UNKNOWN
COMPUTE        NEWDEC = NUMDEC1 + NUMDEC2 + NUMDEC3
ASSIGN MISSING NEWDEC (9999)
PRINT FORMATS  ALPHA (A)/NUMDEC1 TO NUMDEC10, NEWDEC (2)
FREQUENCIES    GENERAL = NEWDEC
READ INPUT DATA
FINISH
```

DEALING WITH SUBFILE STRUCTURES

Some social science data are best analyzed by grouping cases into 2 or more sets of data called "subfiles" in S.P.S.S. Subfiles may be comparative samples from a survey or groups of cases from the same sample based upon the value of some determining variable. All subfiles within a file contain the same variables in the same format but the number of cases may vary. Cases of each subfile are arranged so that all cases in the first subfile appear first in the data deck with the records within cases and cases themselves ordered in exactly the same way they would appear were there no subfile structure. The data can be originally organized in the desired subfiles or the data cases can be sorted or reordered during a run thus creating the correct order of cases.

Subfiles can be created at any time with any set of data and, once defined, are a permanent feature of a file that can be retained or changed on subsequent runs. A subfile structure is defined in S.P.S.S. through the SUBFILE LIST statement which delineates the name of each subfile and the number of cases associated with each. The syntax of the statement is:

```
1              16
SUBFILE LIST   first subfile name (# cases), second subfile name (# cases),...
               last subfile name (# cases)
```

The order that the subfiles are listed must be the order in which they appear in the raw data. The names of subfiles must follow the rules of naming variables in S.P.S.S. with the special restriction that the first four (4) characters of each name must be unique. This statement replaces the N OF CASES statement and is used in conjunction with the creation of a permanent file (SAVE FILE).

Once a subfile structure has been defined, the user must specify exactly how the subfiles are to be processed on any given run. The RUN SUBFILES statement informs the system as to how to process the subfiles. If omitted, the file will be run as a complete file, ignoring the subfile structure. There are four options for processing subfiles.

a. The same calculations are done independently on each subfile (Use the keyword EACH);

```
1              16
RUN SUBFILES   EACH
```

b. A particular subfile or set of subfiles can be selected for independent calculations:

```
1              16
RUN SUBFILES   (subfile name) (subfile name)...
```

c. A number of subfile names may be combined and processed as though each is a unified subfile;

```
1              16
RUN SUBFILES   (subfile name, subfile name...) (subfile name...)
```

d. The subfile structure may be totally ignored (omit the card or use keyword All). This command is placed directly before the first procedural request and remains in force until another RUN SUBFILES statement is encountered in the run.

DEALING WITH ARCHIVE FILES

Although rare, a researcher may need to manipulate over 500 variables for various phases in the analysis. The S.P.S.S. GET ARCHIVE and SAVE ARCHIVE enables the user to create, manipulate and retrieve files of from 500 to 5000 variables. The main purpose of an archive file is to provide unified storage of over 500 variables; however, no more than 500 variables may be referenced (made active) on any one run of S.P.S.S. How, then, is it possible to define and store over 500 variables but not reference them?

Archive files are created from pre-existing system (saved) files by one of three ways: 1) by merging 2 to 5 saved files; 2) by adding variables to a saved file or; 3) by creating new variables from variables within a saved file. The archive file is created and/or accessed by the GET ARCHIVE statement that references the file names of the files that comprise the archive file as well as the "active" variables for the present run. "Active" variables are the only ones that can be used during the run and cannot exceed 500 in number (including any variables created during the run). The SAVE ARCHIVE statement saves the archive file by referencing the file names to be included in the archive and all variables that are to be saved both "active" and "latent" (variables that exist in the files but that were not referenced on the GET ARCHIVE). In this way, files of over 500 active and latent variables may be saved for future access - but on any subsequent GET ARCHIVE runs, no more than 500 variables may be referenced (made active).

The syntax for creating and/or accessing an archive file is:

```
1              16
GET ARCHIVE    FILE= file name  VARIABLES= variable list or ALL/
               FILE= file name  VARIABLES= variable list or ALL/
                .     .    .         .         .       .   .  .
                .     .    .         .         .       .   .  .
               FILE= file name  VARIABLES= variable  list or ALL/
```

Up to five (5) files may be referenced in one archive file. If all the variables from a file are to be made active for the run, the keyword ALL may be used; if only selected variables are to be used, they must be listed in the order in which they reside on the file. Any variable not listed becomes "latent" for the run. Variables from different files should have unique names; if not, only the first variable name will be saved and all others of the same name will be ignored. Each file must have the same number of cases arranged in the same order. This statement directly follows the RUN NAME statement and precedes the FILE NAME that names the archive.

The syntax for saving an archive file for future use is:

```
1              16
SAVE ARCHIVE   archive name  optional archive label/
               FILE= file name  VARIABLES= variable list or ALL/
                .     .    .         .         .       .   .  .
                .     .    .         .         .       .   .  .
               FILE= file name  VARIABLES= variable list or ALL/
```

The archive name and label are both optional and, if omitted, the archive will be saved under the name on the FILE NAME statement. All of the variables that are to be included in the archive file must be referenced on this statement regardless of whether they were referenced on the GET ARCHIVE. If all of the variables from each input file are to be saved, the keyword ALL may be used (the number of variables referenced on this statement may not exceed 5000 including created variables which are automatically saved). This statement directly precedes the FINISH statement.

CONCLUSION

This manual has presented the basic information the beginning researcher needs to know about creating an initial S.P.S.S. program. Organizing the raw data, naming variables, defining their location, type and size, and labeling the variables are all necessary steps to take before conducting even descriptive analysis. This manual also discussed program construction to run temporary programs and to create and access permanent files from the raw data.

Research, however, usually requires more than descriptive analysis and often demands manipulation of the data and higher level statistical procedures. Manual #2 in this series presents all of the statements and commands available in S.P.S.S. that one can add to the basic program to meet specialized research demands. The manual is not written in text form as this one was but presents a listing of non-procedural statements and procedural commands that the researcher can insert in the simple programs he or she learned to write from this manual.

APPENDIX 1

SUMMARY OF S.P.S.S. CONTROL STATEMENTS: PRECEDENCE OF APPEARANCE IN A PROGRAM

The following table is arranged from the first control statements to appear in a program, (highest precedence) to the last to appear (lowest precedence). Any control statement that is included in the chart was either referenced in this manual (denoted by underlining) or is referenced in the second manual in this series. Included in this table is the status of the statement (optional, conditional or required) as well as a general heading of its purpose in a program.

PRECEDENCE	CONTROL WORD	STATUS	PURPOSE
0	EDIT	optional	editing program text
1	NUMBERED	conditional	signals system that col. 72-80 contain id #'s
2	RUN NAME	optional	identifies the run
2	PRINT BACK	optional	specifies content of printed output
2	PAGESIZE	optional	controls # lines per printed page
3	GET FILE	conditional	accesses a saved system file
3	GET ARCHIVE	conditional	accesses a saved system file of over 500 variables
3	MERGE FILES	optional	combines variables from 2 or more system files
4	FILE NAME	conditional	labels an S.P.S.S. file
5	ADD VARIABLES	optional	adds variables to a saved system file
5	ADD CASES	optional	adds new cases to a saved system file
5	ADD DATA LIST	optional	adds variables to a saved system file
5	DATA LIST or VARIABLE LIST	required	identifies variables for inclusion in S.P.S.S. file (temporary or permanent)
6	ADD SUBFILES	optional	adds new subfiles to a saved system file
6	DELETE SUBFILES	optional	deletes subfiles from a saved system file
6	SUBFILE LIST	conditional	specifies the subfile structure for an S.P.S.S. file
7	INPUT MEDIUM	required	signals system where the data resides
7	INPUT FORMAT	conditional	specifies format of raw data-used in conjunction with VARIABLE LIST
8	N OF CASES	required	specifies number of cases in the data
9	MISSING VALUES	optional	assigns a unique value to represent missing information
9	SAMPLE	optional	selects a random sample of cases for analysis
9	SEED	optional	generates the same random sample for multiple runs
10	DO REPEAT	optional	provides a shorthand syntax for similar transformations on many variables
11	COMPUTE	optional	creates new variables
11	IF	optional	creates new variables only if a logical expression is satisfied
11	SELECT IF	optional	selects only those cases for analysis that satisfy a logical expression
11	COUNT	optional	creates an additive index
11	RECODE	optional	changes values of variables
12	END REPEAT	conditional	marks the end of a DO REPEAT

PRECEDENCE	CONTROL WORD	STATUS	PURPOSE
13	WEIGHT	optional	weights cases for analysis
14	VAR LABELS	optional	assigns labels to variables
14	VALUE LABELS	optional	assigns labels for the values of variables
14	DOCUMENT	optional	provides a documentation text to be assigned with a saved system file
15	*SAMPLE	optional	temporarily takes a random sample
16	*COMPUTE	optional	temporarily computes a new variable
16	*IF	optional	temporarily creates a variable conditional upon a logical statement
16	*SELECT IF	optional	temporarily selects cases for analysis conditional upon a logical expression
16	*COUNT	optional	creates a temporary additive index
16	*RECODE	optional	temporarily changes values of variables
17	*WEIGHT	optional	temporarily weights cases
18	ASSIGN MISSING	optional	assigns a unique value to represent missing information for created variables
18	PRINT FORMATS	conditional	specifies the format of decimal and/or alphanumeric variables
19	RUN SUBFILES	conditional	specifies how the subfile structure is to be run
19	RAW OUTPUT UNIT	optional	specifies that output is to be sent to a specific medium
20	LIST CASES	optional	provides a listing of values for specified variables
20	LIST ARCHINFO	optional	provides a listing of values for specified variables in an archive
21	one procedure request	required	activates the system to analyze the data
22	OPTIONS	optional	requests certain options available with a particular subprogram
22	STATISTICS	optional	request additional statistics from a particular subprogram
23	READ INPUT DATA	required	instructs the system to begin reading the raw-data
23	READ MATRIX	conditional	instructs the system to begin reading the matrix input
24	data records	conditional	placed here if data resides on cards - if data is on disk or tape, nothing is placed here
25	additional procedure requests	optional	
26	DELETE VARS	optional	deletes specified variables from a saved system file
26	KEEP VARS	optional	retains specified variables from a saved system file
26	REORDER VARS	optional	changes the sequential position of variables
27	SORT CASES	optional	sorts the cases in a file on specified variables
28	SAVE FILE	conditional	saves the data-definition information and the data as a system file for future access
28	SAVE ARCHIVE	conditional	saves an archive
28	FINISH	required	signals the end of the S.P S.S. program

APPENDIX 2A

GENERAL PROGRAM CONSTRUCTION: TEMPORARY FILE

System control statements to begin the "job," locate the data (if not on cards), and activate S.P.S.S.

Program documentation (RUN NAME, FILE NAME)

Data-definition statement (DATA LIST, VARIABLE LIST)

INPUT MEDIUM

N OF CASES

INPUT FORMAT (if using VARIABLE LIST)

MISSING VALUES

variable modifications and/or transformations

procedural request

READ INPUT DATA

Data records (if on cards)

additional procedural requests

FINISH

system termination statement

APPENDIX 2B

GENERAL PROGRAM CONSTRUCTION: PERMANENT FILE CREATION

System control statements to begin the "job," locate the data (if not on cards), prepare a file for housing the information and activate S.P.S.S.

Program documentation (RUN NAME, FILE NAME)

Data-definition statement (DATA LIST, VARIABLE LIST)

INPUT MEDIUM
N OF CASES
INPUT FORMAT (if using VARIABLE LIST)

MISSING VALUES

Variable modifications and/or transfmorations

Procedural request

READ INPUT DATA
Data records (if data is on cards)

Additional procedural requests

SAVE FILE
FINISH
System termination statement

APPENDIX 2C

GENERAL PROGRAM CONSTRUCTION: ACCESSING A PERMANENT FILE

System control statements to begin the "job," locate the saved system file and activate S.P.S.S.

GET FILE

Variable modifications and/or transformations

Procedural requests

FINISH

System termination statement

APPENDIX 3

REFERENCE LIST

S.P.S.S. References

Statistical Package for the Social Sciences, Second Edition, Norman H. Nie, et. al., New York: McGraw-Hill Book Company, 1975.

S.P.S.S. Update: New Procedures and Facilities for Releases 7 and 8, C. Hadlai Hull and Norman H. Nie, New York: McGraw-Hill Book Company, 1979.

S.P.S.S. Primer, William R. Klecka, et. al., New York: McGraw-Hill Book Company, 1975.

S.P.S.S. Introductory Guide: Basic Statistics and Operations, Marija J. Noruses, New York: McGraw-Hill Book Company, 1982.

A Data Analysis Handbook Using the S.P.S.S. System, Michael J. Cleary, Ph.D. and Robert T. Amsden, Ph.D., Ohio: Advocate Publishing Group, 1980.

S.P.S.S. USER's MANUAL # 2

S.P,S.S. Non-Procedural Statements
and Procedural Commands

Prepared by

Lorna L. Earl

TABLE OF CONTENTS

Introduction.. 1

Chart of S.P.S.S. Commands Found in this Manual............ 3

Non-Procedural Statements.................................. 4

- ADD CASES.. 4
- ADD DATA LIST.. 4
- ADD SUBFILES... 5
- ADD VARIABLES.. 5
- ALLOCATE... 6
- ASSIGN MISSING... 7
- COMMENT.. 7
- COMPUTE.. 8
- COUNT.. 9
- DELETE SUBFILES.. 9
- DELETE VARS.. 10
- DO REPEAT.. 10
- DOCUMENT... 11
- EDIT... 11
- IF... 12
- KEEP VARS.. 13
- LIST ARCHINFO.. 13
- LIST CASES... 14
- LIST ERRORS.. 14
- LIST FILEINFO.. 15
- MERGE FILES.. 15
- NUMBERED... 16
- OPTIONS.. 16
- PAGESIZE... 17
- PRINT BACK... 17
- RAW OUTPUT UNIT.. 18
- READ MATRIX.. 18
- RECODE... 19
- REORDER VARS... 19
- SAMPLE... 19
- SEED... 20
- SELECT IF.. 21
- SORT CASES... 21
- STATISTICS... 22
- TASK NAME.. 22
- WEIGHT... 23
- WRITE FILEINFO... 23

Procedural Commands.. 25

AGGREGATE.................................... 25
ANOVA.. 27
BREAKDOWN.................................... 29
CANCORR...................................... 30
CONDESCRIPTIVE............................... 32
CROSSTABS.................................... 33
DISCRIMINANT................................. 35
FACTOR....................................... 37
FREQUENCIES.................................. 39
GUTTMAN SCALE................................ 40
MULT RESPONSE................................ 41
NONPAR CORR.................................. 42
NPAR TESTS................................... 43
ONEWAY....................................... 44
PARTIAL CORR................................. 46
PEARSON CORR................................. 47
REGRESSION................................... 48
RELIABILITY.................................. 50
REPORT....................................... 52
SCATTERGRAM.................................. 54
SURVIVAL..................................... 56
T-TEST....................................... 58
WRITE CASES.................................. 60

Appendices

Appendix 1... 62
Chart of Arithmetic Operators and Mathematical Functions
Appendix 2... 63
Relational Operators and Logical Operators
Used for IF and SELECT IF Statements
Appendix 3... 64
Reference List

S.P.S.S. USER'S MANUAL #2

S.P.S.S. Introduction Non-procedural Statements and Procedural Commands

This manual presents all of the non-procedural statements and procedural commands (sub-programs) available in S.P.S.S. The first manual in this series presented the basic non-procedural statements to define data and run an S.P.S.S. program - these statements will not be repeated in this text. Precisely, thirty-seven non-procedural statements (any statement other than those that activate a statistical subprogram: RECODE, SORT CASES, DO REPEAT,...) and twenty-three procedural commands (commands that activate a statistical subprogram: FREQUENCIES, REGRESSION, PARTIAL CORR,...) are delineated here.

The format of this manual is as follows:

> Non-procedural statements are presented first in alphabetical order. Each section begins with the name of the statement in capital letters. Directly following this name is a description of the purpose of the statement (Function), then the status of the command for inclusion in a program (Status). The syntax of the statement is presented in a box (Syntax) followed by any special instructions a user should be aware of before using the statement (Special Instructions). Finally, a reference for complete information is provided by Chapter number and page number(s) in the large S.P.S.S. manual Statistical Package for the Social Sciences, Second Edition and, where applicable, chapter number and page number(s) in the new manual S.P.S.S. Update.

Procedural commands are presented beginning on page , also in alphabetical order. The format of this section is identical to the first section with these two exceptions:

1. The Status is replaced by Applicable Variable Types which specifies the appropriate level of data (nominal, ordinal, interval, ratio, or diochotomous) for meaningful use of the subprogram.

2. Two sub-sections are added after the Special Instructions - Options and Statistics. Each present the number and associated alternative available with each subprogram.

The user of this manual must be made aware of the following instructions and considerations in order to effectively use this resource:

1. Any words in all capitals are either commands or keywords involved with the commands and must be used verbatum. Similarly, equal signs, slashes, commas, and parenthesis found in the syntax should be copied as they are illustrated.

2. Statements in brackets are optional; any statement not enclosed in brackets must be included in the syntax for such a statement is the minimal information needed to accomplish the task.

3. Unless otherwise specified, multiple requests can be made on the same command statement by following the first request with a slash (/) and specifying another request using the same syntax.

4. Although the illustrations usually show just one line per statement, lists of variables or specifications can be physically continued on consecutive lines.

5. In the syntatical presentation, certain abbreviations and symbols are used:

   a. dep. var= dependent variable;

   b. indep.var= independent variable;

   c. var(s)= one variable or a list of variables (separated by commas and/or connected by 'TO' for consecutive variables);

   d. #(s)= one number (value) or a series of numbers (values) (separated by commas and/or connected by a THRU for consecutive values);

   e. syntax,...= continue repeating the syntax until finished (nothing follows the end of the statement - no commas, slashes, etc. unless specified); and

   f. keyword(s)= some specified keyword or list of keywords should be used depending upon the user's choice - the keywords are delineated and explained in the Special Instructions section.

6. This document is not meant to provide the complete information for each S.P.S.S. command. An attempt has been made to outline the pre-requisite instructions for the correct use of a command but the user will (at some time) undoubtedly need to refer to the original S.P.S.S. manual for clarification, elaboration and/or information on special circumstances.

Chart of S.P.S.S. Commands Found in This Manual

The following chart is designed to assist the beginning programmer in using this reference manual by outlining general task options in S.P.S.S. and delineating the applicable statement or commands that can be found in this document. Once the general task area is isolated, locate the applicable statements in the manual.

These are the applicable commands for...

...dealing with archives (files of over 500 variables);

LIST ARCHINFO

...data modification and/or transformation:

ASSIGN MISSING
COMPUTE
COUNT
DO REPEAT
IF
RECODE

...program documentation:

COMMENT
DOCUMENT
TASK NAME

...data selection:

SAMPLE
SEED
SELECT IF
WEIGHT

...identification of the form of raw input data:

NUMBERED
READ MATRIX

...editing facilities:

EDIT
LIST ERRORS

...altering an S.P.S.S. system file:

ADD CASES
ADD DATA LIST
ADD SUBFILES
ADD VARIABLES
DELETE SUBFILES
DELETE VARS
KEEP VARS
MERGE FILES
REORDER VARS
SORT CASES

...retrieving information from an S.P.S.S. system file:

LIST CASES
LIST FILEINFO
WRITE FILEINFO

...changing the transpace default for an S.P.S.S. run:

ALLOCATE

...controlling the content and/or form of written output:

PAGESIZE
PRINT BACK
RAW OUTPUT UNIT

...requesting supplementary information provided by a statistical subprogram:

OPTIONS
STATISTICS

NON-PROCEDURAL STATEMENTS

ADD CASES

Function - adds cases to the end of an existing S.P.S.S. file - normally used when the file has no subfile structure.

Status - optional

Syntax -

```
1               16
ADD CASES       #
```

Special Instructions -

1. The number (#) is the number of cases to be added.
2. If used with a file that has a subfile structure, the cases will be added to the end of the last subfile.
3. This statement directly follows the GET FILE command.
4. This statement must also be used in conjunction with an INPUT MEDIUM, INPUT FORMAT and READ INPUT DATA statement to locate and identify the cases to be added.
5. This statement cannot be used in the same run as an ADD SUBFILES statement.

Further Information - Chapter 11 (pp. 157-159)

ADD DATA LIST

Function - adds variables from a raw-data input file to an S.P.S.S. system file. It names the variables, defines their type, location and field width (replaces the ADD VARIABLES and INPUT FORMAT commands).

Status - optional - alternative to ADD VARIABLES and INPUT FORMAT

Syntax -

```
1               16
ADD DATA LIST   FIXED (#)/# var.(s) col.#(s), var(s) col.#(s)...,
                var(s) col.#(s)/# var(s) col.#(s)...var(s)
                col#(s)/# var.(s) col.#(s)...
```

Special Instructions -

1. This statement is identical in format to the DATA LIST command (See Manual #1).
2. This statement directly follows GET FILE.
3. This statement must be used in conjunction with the INPUT MEDIUM and READ INPUT DATA statements (with MISSING VALUES, VAR LABELS and VALUE LABELS (optional).

Further Information - Chapter 11 (pp. 153-155)

ADD SUBFILES

Function - adds new subfiles to the end of an S.P.S.S. system file.

Status - optional

Syntax -

```
1              16
ADD SUBFILES   subfile name (# cases), subfile name (# cases),...
```

Special Instructions -

1. If using this statement for a file without a subfile structure, the existing cases are considered the first subfile with the name of the file assigned to this first subfile.
2. The arrangement of the cases to be added must be identical to the case arrangement in the original file.
3. When naming new subfiles, the first 4 characters must be unique to other subfiles in the file.
4. This statement must be used in conjunction with INPUT MEDIUM, INPUT FORMAT and READ INPUT DATA statements to locate and identify the new subfiles (cases) to be added.

Further Information - Chapter 11 (pp. 157-159)

ADD VARIABLES

Function - adds new variables to an existing S.P.S.S. system file.

Status - optional - alternative to ADD DATA LIST

Syntax -

```
1              16
ADD VARIABLES  var.(s)
```

Special Instructions -

1. There must be the same number of cases arranged in the same order in the raw data file (from which the variable will be added) as in the system file.
2. This command directly follows the GET FILE.
3. The following statements must directly follow the ADD VARIABLES in this order: INPUT MEDIUM, INPUT FORMAT and (after other transformations) READ INPUT DATA.

Further Information - Chapter 11 (pp. 151-153)

ALLOCATE

Function - increases the amount of core space available in the system so as to allow enough "room" for the program to conduct all data modifications.

Status - optional - used only when requiring many data manipulations which necessitates more of the total space alloted for a run to be devoted to storing these changes and less for calculating statistics.

Syntax -

```
1              16
ALLOCATE       TRANSPACE= #
```

Special Instructions -

1. The # indicates the amount of space needed to perform the data modifications.
2. This statement must be placed before any data modification statements for which it applies.
3. ALLOCATE can be used to increase or decrease the amount of space allotted to do data transformations (the default of total space needed for a run is 20,000 "words," 7/8 (or 17,500 words) dedicated to storing data transformations).
4. To calculate the exact amount of space needed for data transformations, take the largest value among the following 3 formulas plus a margin of error:

   TRANSPACE= # of modification cards (RECODE, COMPUTE, COUNT) plus twice the number of IF and SELECT IF cards times 100

   TRANSPACE= # of specific values (or value ranges) in all RECODE and COUNT cards times 25

   TRANSPACE= total # or arithmetic, logical and relational operators plus special functions in all COMPUTE, IF and SELECT IF cards times 12.5

Further Information - Chapter 8 (pp. 111-115)

ASSIGN MISSING

Function - assigns a unique value to a computed variable. If any variable used to create it had missing data thereby creating a 'missing value' for the created variable.

Status - optional

Syntax -

```
1
ASSIGN MISSING var.(s) (#)
```

Special Instructions -

1. If not used, the new variable will be computed using all values of the variables used to create it (ignoring the designated missing value).
2. The '(#)' is the value the new variable will assume and will be recognized as missing data (excluded from future calculations).
3. If a series of IF statements are used to construct a new variable, the ASSIGN MISSING operates independently on each statement.
4. If a new variable is created using the name of an original variable that already has a missing value, the ASSIGN MISSING will override the original missing value.
5. This statement must be placed anywhere after the transformations and before the first statistical procedure (unless the variable was created temporarily when it would follow the computation and directly preceded the task request).

Further Information - Chapter 8 (pp. 115-120)

COMMENT

Function - performs program documentation.

Status - optional

Syntax -

```
1              16
COMMENT          any comment
```

Special Instructions -

1. This statement may appear anywhere in the program except for in the middle of another control card and its continuations, between READ INPUT DATA and first data record or between the STATISTICS and OPTIONS statements.
2. This statement is not processed but is printed in the program.
3. The comment may be continued on several cards (col. 16-80).

Further Information - Chapter 6 (p. 75)

COMPUTE

Function - creates a new variable which is an arithmetic function of one or more existing variables.

Status - optional

Syntax -

```
1              16
COMPUTE        new variable name = arithmetic expression
```

Special Instructions -

1. The arithmetic expression can consist of existing variables, constants and arithmetic and special mathematical functions (see Appendix 1 for the S.P.S.S. symbols used).
2. The computed variable name can be the same as a pre-existing variable name but the "old" value(s) will be lost and the variable will take on the "new" created value(s).
3. In constructing the arithmetic expression, these rules must be followed:
   a. No two operators can appear contiguously;
   b. Never assume an operator present;
   c. Parenthesis are used to determine the order of operation and must balance; and
   d. If no parenthesis are used, the operations will be performed in this order: special functions are done first, then exponentiation, then multiplication and division and lastly, addition and subtraction from left to right.

Further Information - Chapter 8 (pp. 96-101)

COUNT

Function - computes a new variable by summing the number of occurences of a specific value (or list of values) for the chosen variables - creates an additive index.

Status - optional

Syntax -

```
1              16
COUNT          variable name= var.(s) (#) var.(s) (#)...var.(s) (#)
```

Special Instructions -

1. For each case, the number of times the specified value(s) for the specified variable(s) occurs, the value of the new variable is incremented by 1.
2. The value list can be: a single value, a series of values separated by a comma; a range of values (using the keywords THRU, LOWEST and HIGHEST); or a combination of the three.
3. Variables may be listed more than once thereby counting them more than once.
4. Temporary (*) COUNT statements are available.

Further Information - Chapter 8 (pp. 107-110)

DELETE SUBFILES

Function - deletes all cases in one or more subfiles in an existing system file.

Status - optional

Syntax -

```
1               16
DELETE SUBFILES subfile name, subfile name,...
```

Special Instructions -

1. This statement appears after the GET FILE and automatically deletes the specified subfiles (the subfiles are not available for processing at any time during the run).
2. This statement can be used with ADD SUBFILES in the same run (the order of appearance does not matter).

Further Information - Chapter 11 (p. 159)

DELETE VARS

Function - deletes the specified variable(s) from a system file thereby retaining only the desired set of variables (used in conjunction with SAVE FILE).

Status - optional

Syntax -

```
1              16
DELETE VARS    var.(s)
```

Special Instructions -

1. This statement is placed directly before the SAVE FILE.
2. The variables marked for deletion may be used in the same run that eventually removes them.

Further Information - Chapter 11 (pp. 149-150)

DO REPEAT

Function - reduces control-statement preparation by repeating the same transformation(s) on multiple variables through the use of stand-in variables and one replica of the desired transformation.

Status - optional

Syntax -

```
1              16
DO REPEAT      stand-in var1= variable list1/
               stand-in var2= variable list2/
                  .    .   .        .       .
                  .    .   .        .       .
                  .    .   .        .
               stand-in varn= variable listn/
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
|  sequence of control cards making one copy of the desired |
|  transformation using the stand-in variable names         |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
END REPEAT
```

Special Instructions -

1. Stand-in variable names must be distinct from all other variable names in the DO REPEAT (since stand-in names are "forgotten" after the END REPEAT, the same stand-in names may be used in another DO REPEAT).

2. Each variable list must contain the same number of variables and must be in the same order (since all the first variables in the lists are inserted in the replica, then all the second variables, etc.).
3. Any variable present in the replica must have an associated variable list - even created variables.
4. Included in the variable list could be a series of constants to be used in the calculation.
5. The valid control statements in a DO REPEAT are:

   IF
   COMPUTE
   RECODE
   COUNT
   SELECT IF
   MISSING VALUES
   ASSIGN MISSING
   *IF
   *COMPUTE
   *RECODE
   *COUNT
   *SELECT IF

<u>Further Information</u> - Chapter 8 (pp. 121-125)
Chapter 7 (pp. 165-167) in S.P.S.S. Update Manual

DOCUMENT

<u>Function</u> - retains the text of documented information as part of an S.P.S.S. system file that can be retrieved by the use of a LIST FILEINFO statement (used with a SAVE FILE).

<u>Status</u> - optional

<u>Syntax</u> -

```
1              16
DOCUMENT       any text ____________
                        ____________
```

<u>Special Instructions</u> -

1. Only <u>one</u> DOCUMENT may be used in a run.
2. The statement may be placed anywhere before the first statistical procedure.

<u>Further Information</u> - Chapter 6 (p. 76)

EDIT

<u>Function</u> - inspects control cards for syntax errors and correct variable names without reading the data or performing the requested statistical analysis.

<u>Status</u> - optional

<u>Syntax</u> -

```
1              16
EDIT
```

Special Instructions -

1. This statement must be the very first statement in the program.
2. Do not include the data records (or file specifications on where the data is located) in the program.
3. EDIT examines each statement individually, the order of statements or completeness of the program is not evaluated.

Further Information - Chapter 13 (pp. 176-180)

IF

Function - performs variable transformations contingent upon specified logical conditions.

Status - optional

Syntax -

```
1               16
IF              (arithmetic-expression relational arithmetic-expression)
                                        operator
                computed variable= arithmetic-expression
```

Special Instructions -

1. For each case the expression in parenthesis is evaluated to see if it meets the criteria. If so, the computed variable is defined as specified; if not, the variable remains undefined.
2. An arithmetic expression can contain variable names, constants, or arithmetic operators and mathematical functions - all follow the same rules for processing (see COMPUTE, Special Instruction #3).
3. The arithmetic expressions are joined (or compared) by a relational operator (e.g. "equal to," "less than;" see Appendix 2 for the S.P.S.S. symbols used).
4. The relational operators are processed before logical operators (See Appendix 2) and after arithmetic operators.
5. Only one transformation can be done on an IF statement (cannot use a slash and then specify another logical expression - another IF statement would be needed).
6. Temporary (*) IF statements are available.

Further Information - Chapter 8 (pp. 101-107)

KEEP VARS

Function - retains only specified variables from an S.P.S.S. system file (used in conjunction with SAVE FILE).

Status - optional

Syntax -

```
1              16
KEEP VARS      var.(s)
```

Special Instructions -

1. All variables in the file can be used during the run which eventually keeps only certain variables.
2. This statement appears directly before the SAVE FILE.

Further Information - Chapter 11 (pp. 149-151)

LIST ARCHINFO

Function - prints the variable names and data definition information on all input files in an archive.

Status - optional

Syntax -

```
1              16
LIST ARCHINFO  keyword
```

Special Instructions -

1. There are two available keywords (only one of which can be used):
    a. VARLIST - will list all the variable names in their sequential positions (both latent and active variables)
    b. COMPLETE - lists the variable names, variable and value labels, missing values, print format and relative position (active variables marked with *).
2. This statement may be placed anywhere after the GET ARCHIVE and will provide information on any updated changes to that point.

Further Information - Chapter 12 (pp. 173-174)

LIST CASES

Function - prints a listing of values of specific variables for the first 'n' cases in a file or in each subfile group.

Status - optional

Syntax -

```
1              16
LIST CASES     CASES=#/VARIABLES= var.(s)
```

Special Instructions -

1. Since this is not a statistical procedure, it must be used in conjunction with a procedural (statistical) command.
2. The LIST CASES must be placed directly before a command that activates the file (e.g., FREQUENCIES).
3. The '#' is the number of cases to be listed (not to exceed the total number of cases in a file). If omitted, only the first 10 cases will be listed.

Further Information - Chapter 10 (pp. 137-138)

LIST ERRORS

Function - provides a complete listing of S.P.S.S. error messages that could be encountered in any S.P.S.S. program.

Status - optional

Syntax -

```
1              16
LIST ERRORS
```

Special Instructions -

1. The result of this statement is a list of each S.P.S.S. error number and explanation (approximately 30 printed pages).
2. Can appear anywhere in a program except after SORT CASES or SAVE FILE (or alone in a program).

Further Information - Chapter 15 (p. 219) in S.P.S.S. Update Manual

LIST FILEINFO

Function - provides a listing of variable names and data - definition information on an S.P.S.S. system file.

Status - optional

Syntax -

```
1              16
LIST FILEINFO  keyword(s)
```

Special Instructions -

1. The available keywords are:
    - a. VARLIST - lists all variables in the file according to sequential position
    - b. SORTVARS - alphabetically sorts list of variables and their sequential position
    - c. VARINFO - lists print formats, status and missing values of all variables
    - d. LABELS - lists variable and value labels associated with each variable
    - e. SUBDIRECTORY - lists the current subfile names and the number of cases in each
    - f. COMPLETE - reports all data-definition data associated with each variable
    - g. DOCUMENTS - lists all information entered in the file via the DOCUMENT card
2. This statement may be placed anywhere following the GET FILE.

Further Information - Chapter 10 (pp. 134-137)

MERGE FILES

Function - creates a unified system file by merging all or a subset of variables from two to five existing files of the same number of cases.

Status - optional - alternative to GET ARCHIVE when dealing with fewer than 500 variables

Syntax -

```
1              16
MERGE FILES    FILE= file name VARIABLES= var.(s) or ALL/
               FILE= file name VARIABLES= var.(s) or ALL/
                 :     :    :       :         :    :  :
               FILE= file name VARIABLES= var.(s) or ALL/
```

Special Instructions -

1. Each file must have a unique name with the same number of cases arranged in the same order.
2. The subfile structure and case weights will be taken from the first file only.
3. This statement follows the RUN NAME (in place of GET FILE).
4. The variables on the variable list must be entered in the order they reside on the file and must have unique names (if not, the variables with the same name will be taken only once - from the file in which they were it was first encountered). If all variables in a file are desired, use the keyword ALL.
5. To save a merged file, place a FILE NAME after the MERGE FILES and a SAVE FILE before the FINISH statement (a saved merged file will act the same as a regular saved file that can be retrieved with a GET FILE as long as it contains less than 500 variables).

Further Information - Chapter 11 (pp. 155-157)

NUMBERED

Function - causes columns 73-80 of all control statements and their continuations to be ignored so that these columns can be used for sequencing or identification.

Status - optional

Syntax -

```
1              16
NUMBERED       YES
```

Special Instructions -

1. The 'YES' is optional - if this card is not used, the columns will not be ignored.

Further Information - Chapter 6 (pp. 76-77)

OPTIONS

Function - provides the system with further information to be used in controlling a calculation activated by a procedure (statistical subprogram).

Status - optional

Syntax -

```
1              16
OPTIONS        list of numbers
```

Special Instructions -

1. In each description of a subprogram, a section outlines the available options within the analysis and their associated numbers.
2. The numbers are separated by commas and, if all options are desired, each number must be listed.
3. This command follows the procedure statement and can come before or after the STATISTICS request.

Further Information - Chapter 5 (p. 66)

PAGESIZE

Function - specifies the number of lines to be printed on each page of output before skipping to the next page.

Status - optional

Syntax -

```
1              16
PAGESIZE       # or NOEJECT
```

Special Instructions -

1. The default is 55 lines per page.
2. NOEJECT prints the lines continuously without skipping lines - a dashed line separates logical pages of output.

Further Information - Chapter 6 (p. 74)

PRINT BACK

Function - controls the printed output of the user's control cards by limiting certain information.

Status - optional

Syntax -

```
1              16
PRINT BACK     keyword
```

Special Instructions -

1. Normally (if not used) the output will contain a printed list of all control statements used and a format statement of the variables in the file.
2. Three keywords are available:
    a. NO - will print neither
    b. CONTROL - will print only the control statements
    c. FORMAT - " " " " format statement
3. This statement is placed after the RUN NAME.

Further Information - Chapter 6 (pp. 74-75)

RAW OUTPUT UNIT

Function - directs output data to a chosen logical unit instead of defaulting each data set to be written on one logical unit (defaults to 9).

Status - optional

Syntax -

```
1              16
RAW OUTPUT UNIT #
```

Special Instructions -

1. The valid numbers to use are 9, 15, 16, 17, 18, 19 and 20 (raw data output units).
2. Each RAW OUTPUT UNIT statement must directly precede the task that produces raw data output. Several such commands can be present in one program.
3. A RAW OUTPUT UNIT will remain in effect until the next one is encountered.
4. For each of these statements, a corresponding system file-equate* are to be written on the default unit 9 meaning this command need not be present, one file statement must be included to identify the type of logical unit the data should be written on) (see Manual #3 for an explanation of file-equate statements).

Further Information - Chapter 5 (pp. 69-70)

READ MATRIX

Function - instructs the system to begin reading correlation matrices or other calculated data instead of raw data.

Status - required when using matrices for subprograms that optionally allow matrices as input data (e.g., FACOTR, REGRESSION, PARTIAL CORR)

Syntax -

```
1              16
READ MATRIX
```

Special Instructions -

1. The following control statements must be present when using this command: VARIABLE LIST, INPUT MEDIUM, N OF CASES, and the OPTIONS card for the subprogram which indicates that this alternative to raw data input will be used.

Further Information - Chapter 19 (pp. 308-312)

*Statement must be present in the program

RECODE

Function - recodes any or all variables in a file so that one, some or all values are set equal to a new value set.

Status - optional

Syntax -

```
1              16
RECODE         var.(s) (old value(s)= new value) (old value(s)= new value)...
```

Special Instructions -

1. Any variable type can be recoded (including blanks: (BLANK = value)) (alphanumerics require the keyword CONVERT after the value recodes).
2. There may be several original values listed but only one new value.
3. In the list for the original values, these keywords may be used: THRU, LOWEST (LO), HIGHEST (HI) or ELSE (for residual values).
4. Once a variable has been recoded, the original value is lost.
5. Temporary (*) RECODE is available.

Further Information - Chapter 8 (pp. 90-96)

REORDER VARS

Function - reorders variable positioning in a system file (used in conjunction with SAVE FILE).

Status - optional

Syntax -

```
1              16
REORDER VARS   var.(s)
```

Special Instructions -

1. This statement is placed directly before the SAVE FILE.
2. The order in which variables are listed on the command defines how they will reside in the new file.
3. Variables not on the statement will be placed at the end of the file in their original order.

Further Information - Chapter 11 (pp. 160-162)

SAMPLE

Function - draws a random sample of data cases from a file.

Status - optional

Syntax -

```
1               16
SAMPLE          decimal #

                   or

                # FROM #
```

Special Instructions -

1. The decimal number must be positive and represents the percentage of the sample to take (e.g., .25 would take a random 25% or fourth of the data cases to analyze).
2. A random sample of exact size may be specified by the second syntax where the first number is the exact number of cases to be included in the sample and the second number is the exact number of cases from which the sample is to be drawn.
3. Temporary (*) SAMPLE is available.

Further Information - Chapter 9 (pp. 127-128)
Chapter 7 (pp. 157-159) in S.P.S.S. Update Manual

SEED

Function - allows the reproduction of random samples generated by SAMPLE.

Status - optional

Syntax -

```
1               16
SEED            a larger integer
                or
                PRINT
```

Special Instructions -

1. The random sample generated in S.P.S.S. is based upon a seed (selected at the time of day the program is run).
2. The large number (six to eight digits) specifies a seed number and if a set of procedure commands (including SAMPLE) are run with the same number on this card, the same random sample will be drawn.
3. The keyword PRINT causes the actual seed number to be printed so that it can be reproduced later.
4. The SEED statement may be placed anywhere in the program (usually before the SAMPLE).

Further Information - Chapter 7 (pp. 158-159) in S.P.S.S. Update Manual

SELECT IF

Function - processes only those cases for which certain criteria are met.

Status - optional

Syntax -

```
1              16
SELECT IF      (logical expression)
```

Special Instructions -

1. The logical expression consists of variable names, relational and logical operators, constants, and arithmetic and special functions.
2. If two or more SELECT IF commands are used, together they are connected by an implied AND - (all specifications in each SELECT IF must be met for a case to be processed).
3. The statement operates in the same way as the IF statement (see IF page   ).
4. A temporary (*) SELECT IF is available.

Further Information - Chapter 9 (pp. 128-219)

SORT CASES

Function - reorders the sequence of cases in a file on the values of some criteria variable(s) (used in conjunction with SAVE FILE).

Status - optional

Syntax -

```
1              16
SORT CASES     var.(s) (A or D) var.(s) (A or D)...
```

Special Instructions -

1. A file may be sorted on from one to sixty four variables.
2. The values of variables can be sorted from ascending (A) or descending (D) order - ascending order is a default therefore not required in the syntax.
3. This command may appear before any procedure and can be used more than once. Permanent sorts are created by saving the final sort mentioned.
4. This command can automatically create subfiles that contain the sorted cases - the particular combinations of the values of the variables will each represent a subfile. (If no names are specified, the first grouping of cases will be called S1, the second S2 and so on)

```
1              16
SORT CASES     var.(s)/SUBFILES=subfile name1,
               subfile name2,...
```

5. The original order of cases can be preserved plus subfiles of sorted cases created by inserting SEQNUM after the list of variables (in ascending order only).

```
1             16
SORT CASES    var.(s) SEQNUM/SUBFILES
```

Further Information - Chapter 11 (pp. 162-165)
Chapter 10 (pp. 180-183) in S.P.S.S. Update Manual

STATISTICS

Function - enables the selection of a variety of statistics that accompany the calculations done in a statistical subprogram.

Status - optional

Syntax -

```
1             16
STATISTICS    number list or ALL
```

Special Instructions -

1. The numbers in the list are separated by commas.
2. The numbers and their associated statistics are outlined in the section describing the particular subprogram.
3. This statement appears directly after the procedure to which it applies but may appear before or after OPTIONS.

Further Information - Chapter 5 (pp. 66-67)

TASK NAME

Function - documents specific procedures within a program.

Status - optional

Syntax -

```
1             16
TASK NAME     any label up to 64 characters
```

Special Instructions -

1. This command usually appears before the procedure it is documenting and the label is printed at the top of the appropriate pages of output.
2. This label will continue to be printed on the output until another TASK NAME is encountered.

Further Instructions - Chapter 6 (p. 73)

WEIGHT

Function - weights cases more or less heavily than other cases in a file so that the appropriate weighting factor is used in statistical analysis.

Status - optional

Syntax -

```
1              16
WEIGHT         variable name
```

Special Instructions -

1. The variable name represents the mixed number by which to multiply the values of variables for the cases in a sample.
2. The weighting factor can be precalculated and entered on the original raw data or it can be calculated in the program.
3. This command appears after all permanent data-modification cards and before the first procedure.
4. Temporary (*) WEIGHT is available.

Further Information - Chapter 9 (pp. 129-131)

WRITE FILEINFO

Function - writes any or all data-definition information which currently resides in a system file on a medium of the user's choice (disk, tape or cards) (used in conjunction with GET FILE).

Status - optional

Syntax -

```
1              16
WRITE FILEINFO VARIABLES= variable list/
               list of keywords or ALL/
               CHAR=BCD or EBCD/
               NUMBERED
```

Special Instructions -

1. The 'VARIABLES=' is optional. If not used, information on every variable will be output; if used, the variables may be entered in any order thereby controlling the way the variables appear on the VARIABLE LIST that is output.
2. The list of keywords available can appear in any order and any combination or the keyword ALL will give every piece of information on the specified variables. The keywords are:
    a. FILENAME- writes a FILE NAME for the file
    b. VARLIST- writes a VARIABLE LIST for the specified variables
    c. SUBFLIST- writes a SUBFILE LIST for the file
    d. NCASES- writes an N OF CASES
    e. MISVALS- writes a MISSING VALUES for the variables
    f. PRNTFMTS- writes a PRINT FORMATS for the variables
    g. VARLABS- writes a VAR LABELS
    h. VALLABS- writes a VALUE LABELS
    i. DOCUMENTS-writes a DOCUMENTS and all the text associated with it
3. This statement produces program cards of 80 characters long for each keyword specified.
4. The 'CHAR=' allows the user the option of character sets produced by IBM 029 keypunches (EBCD) or by IBM 026 keypunches (BCD). If omitted, the characters will be written in the character set normally used by the computer being used.
5. NUMBERED optionally allows sequence numbers to appear in columns 73-80 (if used, must use the statement NUMBERED when subsequently running the deck).

Further Information - Chapter 10 (pp. 145-148)

PROCEDURAL COMMANDS

AGGREGATE

Function - summarizes the characteristics of a "group" of cases where each group is composed of the individual cases that are members of a particular higher-level unit. It creates a raw data output file containing the descriptive statistics for chosen variables for each "group."

Applicable Variable Types - any

Syntax -

```
1              16
AGGREGATE      GROUPVARS= var.(s)/
               VARIABLES= var.(s)/
               AGSTATS= keyword(s)/
               RMISS=
```

Special Instructions -

1. Two types of aggregated files are available:
   a. "True" - only the grouped information is stored - individual cases are lost (default); and
   b. "Compositional" - the original case data is retained as well as the grouped information.
2. Raw data must be sorted (SORT CASES) prior to use of this sub-program (all members of each group must reside contiguously in the file).
3. A group is defined as all cases in a file which have identical values for all of the grouping variables named in the first statement (cases with missing values are excluded).
4. RMISS is optional but recommended because it assigns a chosen numeric value for incalculatable statistics.
5. The available statistics for an aggregated file are:

   VALIDN
   SUM
   MEAN
   SKEW
   KURT
   PCTGT
   SD
   MAX
   MIN
   PCTLT
   PCTBTN

Options:

1. Inclusion of missing data
2. List wise deletion of missing data
3. Prints the values of the grouping variables
4. Generates a "compositional" aggregated file

Statistics -

1. Reports group id. numbers, total number of cases, and values for the grouping variables
2. Same as Statistic 1 plus reports the exact contents of each case
3. Same as Statistic 2 but only for the first 10 cases

Further Information - Chapter 15 (pp. 203-217)
Chapter 15 (p. 213) of S.P.S.S. Update Manual

ANOVA

Function - provides for a one-way to five-way analysis of variance and/or covariance for factorial designs. Determines the effect of one variable (factor) or the simultaneous effect of two or more variables (factors) on another variable.

Applicable Variable Types -

dependent variable: interval or ratio.
independent variables: any level (nominal or ordinal are called factors, interval or ratio are called covariates)

Syntax - "n-way analysis of variance" (use of nominal or ordinal independent variables)

```
1              16
ANOVA          dep. var.(s) BY indep. var.(s) (min., max.) indep. var.(s)
               (min., max.)... indep. var(s) (min., max.)
```

- "analysis of covariance" (use of at least one interval level independent variable)

```
1              16
ANOVA          dep. var.(s) BY indep. var.(s) (min., max.)...
               WITH covariate indep. var.(s)...
```

Special Instructions -

1. The independent variables listed on the command are the factors and/or covariates that are chosen to examine the variance in the dependent variable (the factorial design).
2. The output statistics provided are: the sum of squares, degrees of freedom, mean square, F value and level of significance of the F value for each independent variable separately and for the independent variables simultaneously.
3. The (min., max.) denotes the smallest and largest values of the independent variables to use in the analysis. Cases with values outside this range will be ignored.

Options -

1. Ignores missing values
2. Suppresses value labels
3. Ignores two-way or higher interactions among factors
4. Ignores three-way " " " "
5. " four-way " " " " "
6. " five-way interactions among factors
7. Processes covariates concurrently with the main effects for the non-metric factors
8. Processes covariates after the main effects for the non-metric factors
9. Regression approach
10. Heirarchical approach

Statistics -

1. Multiple classification analysis
2. Unstandardized partial regression coefficients for the covariates are printed

Further Information - Chapter 22 (pp. 298-422)
Chapter 13 (pp. 206-208) in S.P.S.S. Update Manual

BREAKDOWN

Function - examines central tendency by calculating and printing the sums, means, standard deviations and variances of a dependent variable among sub-groups of cases in the file (represented by categories of the independent variables).

Applicable Variable Types -

dependent variable: numeric for which a mean is a reasonable measure of central tendency (usually interval or ratio)

independent variable: any level but must be classified into a small number of discrete categories.

Syntax -

```
1               16
BREAKDOWN       TABLES= dep. var.(s) BY indep. var.(s) BY indep. var.(s)...
                BY indep. var.(s)
```

```
1               16
BREAKDOWN       VARIABLES= var.(s) (min., max.), var.(s) (min., max.)...
                var.(s) (min., max.)/
                TABLES= dep. var.(s) BY indep. var.(s)...
```

```
1               16
BREAKDOWN       VARIABLES= var.(s) (min., max.).../
                CROSSBREAK= dep. var.(s) BY indep. var.(s)...
```

Special Instructions -

1. The maximum number of independent variable sets that can be entered is 5 (or maximum of 5 'BY' keywords in one table request).
2. There are two operating modes: general (first illustration) and integer (second illustration - for each variable, lowest and highest values to be calculated must be defined).
3. The third illustration presents a hybrid of BREAKDOWN and CROSSTABS called CROSSBREAK that prints the BREAKDOWN table in cross-tabular form. Only for the integer mode.

Options -

1. Inclusion of missing data
2. Exclusion of missing data for dependent variables only
3. Suppresses labels
4. Tree-diagram style output (general mode only)

Statistics -

1. One-way analysis of variance
2. Test of linearity on the one-way breakdown

Further Information - Chapter 17 (pp. 249-266)
Chapter 15 (pp. 214-215)

CANCORR

Function - performs canonical correlation analysis on the data. Accounts for the maximum amount of the relationship between two sets of variables.

Applicable Variable Types -

dependent variables- any level
independent variables- any level

Syntax -

```
1              16
CANCORR        VARIABLES = var.(s)
               RELATE = (parameters) var. set 1 WITH var. set 2/
               CANVAR = mdrp or CANVAR
```

Special Instructions -

1. Input for this sub-program can be either raw data, a system file or a correlation matrix.
2. Raw data file outputting of the individual case scores on the canonical variates can be done on cards, tape or disk.
3. The '(parameters)' is optional and allows the specification of 3 parameters for computing the canonical variates: the number of canonical variates to be computed(n), the significance level for ceasing to compute further canonical variates (F) and the tolerance level to detect problems in the correlation matrix. The default is (5, .05, .00001).
4. The missing value replacement proportion (mdrp) is a decimal number between 0.0 and 1.0 (used if pair-wise deletion of missing values is desired).
5. Since the first step in this type of analysis is to create a correlation matrix of the variables to be included, the deck must contain a VARIABLE LIST, INPUT MEDIUM, N OF CASES and READ MATRIX statement.
6. When using matrix input, the CANVAR keyword cannot be used.
7. The output from the subprogram will produce two types of information:
    a. a summary table (magnitudes of the eigenvalues, the canonical correlations, and the statistical significance of the correlations) and
    b. two canonical variate matrices.

    If CANVAR is used, a raw output data file containing canonical covariate scores will be produced.

Options -

1. Inclusion of missing data
2. Pair-wise deletion of missing data
3. Computes a covariance matrix
4. Matrix input
5. Computes matrix about the origin
6. Continues computations if either variable is linearly dependent
7. Continues computations even if matrix is not positive definite
8. Outputs a simple correlation matrix
9. Input correlation matrix is indexed by the VARIABLE LIST card
10. Weighted estimate of missing data in computing canonical scores

Statistics -

1. Means and standard deviations
2. Correlation matrix
3. Printing of correlation matrix when non-computable correlations are encountered
4. Canonical variate scores

Further Information - Chapter 25 (pp. 515-527)
Chapter 15 (p. 215) in S.P.S.S. Update Manual

CONDESCRIPTIVE

Function - describes the basic distributional characteristics of variables in a sample. Computes summary statistics on measures of central tendency for continuous variables.

Applicable Variable Type - variables with numerous categories in which a mean is a reasonable measure of central tendency (usually interval).

Syntax -

```
1               16
CONDESCRIPTIVE  var.(s)
```

Special Instructions -

1. To get only selected descriptive statistics for the variables listed, the STATISTICS card must be used. If omitted, all of the available statistics will be printed.

Options -

1. Inclusion of missing data
2. Suppresses variable labels
3. Creates a raw data output file containing Z scores
4. Prints a reference dictionary

Statistics -

1. Mean
2. Standard error
5. Standard deviation
6. Variance
7. Kurtosis
8. Skewness
9. Range
10. Minimum
11. Maximum
12. Sum of a variable over all cases

Further Information - Chapter 14 (pp. 181-193)
Chapter 15 (pp. 215-216) of S.P.S.S. Update Manual.

CROSSTABS

Function - produces a two-way to n-way joint frequency distributions of cases as defined by the categories of the variables involved in the form of a table.

Applicable Variable Types -

dependent: any level if limited number of discrete categories
(usually nominal or ordinal)
independent: same

Syntax -

```
1              16
CROSSTABS      TABLES= dep. var.(s) BY indep. var.(s) BY indep. var.(s)
               BY...indep. var.(s)
```

```
CROSSTABS      VARIABLES=var.(s) (min., max.), var.(s) (min., max.).../
               TABLES= dep. var.(s) BY indep. var.(s)...
```

Special Instructions -

1. CROSSTABS operates under two modes: general (first illustration) and integer (second illustration - must specify upper and lower limits of the values for each variable).
2. The first variable is the dependent variable, the second is the independent variable and all other 'BY indep. var' sequences are control variables. Up to 9 "...indep. var.(s) BY indep. var.(s)..." sequences can be requested on a single table request (general mode) (integer mode can handle up to 7).
3. If no STATISTICS card is used, the sub-program will only print the requested table(s) with this information: raw frequency for each cell, column percentage, row percentage, total percentage and marginals.
4. Only 16 out of the maximum of 20 characters of the value labels for a variable will be printed on the output.

Options -

1. Inclusion of missing data
2. Suppression of labels
3. Deletion of row percentages
4. " " column "
5. " " total "

The following options only available in the integer mode:

6. Value labels suppressed, variable labels printed
7. Inclusion of missing data in table but not in calculations
8. Causes the row variable values to be printed from highest to lowest instead of lowest to highest
9. Produces an index of tables
10. Raw data output file of non-empty cells
11. " " " " " all defined cells
12. Suppresses printed output

Statistics -

1. Chi-square (Fisher's exact test for 2 X 2 tables with less than 21 cases)
2. Phi (2 X 2 tables) or Cramer's V (large tables)
3. Contingency coefficient
4. Lambda, symmetric and asymmetric
5. Uncertainty coefficient, symmetric and asymmetric
6. Kendall's tau b
7. Kendall's tau c
8. Gamma (Partial and zero-order gamma for larger than 2 variable tables)
9. Somer's D, symmetric and asymmetric
10. Eta
11. Pearson's product moment correlation

Further Information - Chapter 6 (pp. 218-248)
Chapter 12 (pp. 200-205) of S.P.S.S. Update Manual

DISCRIMINANT

Function - performs discriminant analysis either by entering all variables directly into the analysis or through a variety of stepwise methods selecting the "best" set of discriminating variables. Statistically distinguishes between two groups of cases along user chosen variables (discriminating variables) that measure characteristics on which groups are expected to differ.

Applicable Variable Types - dependent: ordinal interval or high
independent: any level

Syntax -

```
1               16
DISCRIMINANT    GROUPS= var. (min., max.)/
                        or
                        SUBFILES/
                VARIABLES= var.(s)/
                ANALYSIS= var.(s) (inclusion level),
                          (inclusion level).../
```

Special Instructions -

1. Only one group variable is allowed and its values must be integer.
2. All variables in this analysis must be numeric and any variable being analyzed must be defined on the "VARIABLES=" statement.
3. On the "ANALYSIS=" statement, the "(inclusion level)" is needed only for stepwise analysis. If excluded and doing stepwise, all variables will compete equally for entry in the analysis. The function of this level (integer from 0 to 99) is to direct the order of entry of the variables into the analysis.
4. For direct analysis, the variables will be entered concurrently.
5. In order to request a direct analysis, this statement must follow the "ANALYSIS=" statement:

```
16
METHOD = DIRECT
```

Options -

1. Inclusion of missing data
2. Inclusion of cases with missing data during classification
3. Suppression of step-by-step output
4. Suppression of stepwise summary table
5. Printing of classification results table
6. Printing of discriminant scores and classification information
7. Printing of a single plot of cases
8. Printing of a separate plot for each group
9. Suppression of classification phase features for cases initially unclassified
10. Printing of territorial map

11. Printing of unstandardized discriminant function coefficients
12. Printing of classification functions
13. Rotation of the discriminant functions
14. Use individual group covariance matrices for classification
15. Output matrix materials
16. Matrix input
17. Output discriminant scores
18. Output membership probabilities for all groups
19. Output actual group and classified group members

Statistics -

1. Means
2. Standard deviations
3. Pooled within-groups covariance matrix
4. Pooled within-groups correlation
5. Matrix of pairwise F-ratios
6. Univariate F-ratios
7. Test for equality of group covariance matrices
8. Group covariance matrices
9. Total covariance matrices

Further Information - Chapter 23 (pp. 434-467)
Chapter 11 (pp. 184-199) of S.P.S.S. Update Manual

FACTOR

Function - performs a variety of factor-analytic techniques. Given an array of correlation coefficients for a set of variables, patterns of relationships are detected and source variables (factors) are derived which account for inter-relations in the data thereby summarizing or reducing the data. There are 3 steps in the procedure:

1. preparation of correlation matrix
2. extraction of original factors
3. rotation to terminal factors

Applicable Variable Types - any

Syntax -

```
1              16
FACTOR         VARIABLES= variable list/
             ┌ TYPE= keyword/                                        ┐
             │ DIAGONAL= value list/                                 │
             │                                                       │
             │ NFACTORS= whole #/                                    │
             │                                                       │
             │ MINEIGEN= decimal #/                                  │
             │                                                       │
             │ ITERATE= whole #/                                     │
             │                                                       │
             │ STOPFACT= decimal #/                                  │
             │ ROTATE= keyword/                                      │
             │ FACSCORE/                                             │
             │    or                                                 │
             └ FACSCORE= proportion of missing data allowed/         ┘
```

Special Instructions -

1. Input for this procedure may be raw data, a system file, a correlation matrix or a factor matrix.
2. The only necessary section of the syntax is "VARIABLES=" - all other sections have defaults and can be omitted if the default is desired.
3. Five methods of factoring are available (keywords for the "TYPE=" statement are in parenthesis):
   - a. principal factoring without iteration (PA1)
   - b. " " with " (PA2) (default)
   - c. Rao's canonical factoring (RAO)
   - d. alpha factoring (ALPHA)
   - e. image factoring (IMAGE)
4. There are 3 common features to all of the 5 extracting methods:
   - a. All factors are imposed to be orthogonal.
   - b. Factors are arranged in the order of their importance (most important first).
   - c. The first factor tends to be general (significant loading on every variable) while subsequent extracted factors tend to be bipolar.

5. Rotation of factors is not manditory but suggested because it simplifies the factor structure. There are 3 orthogonal methods (uncorrelated factors)
    a. QUARTIMAX
    b. VARIMAX (default)
    c. EQUIMAX

    and 1 oblique method (correlated factors):
    d. OBLIQUE
6. The other statements presented in the syntax allow complete user control of factoring specifications. These are the defaults:
    a. DIAGONAL= 1.0 (only if PA1 method is used - alters diagonal in correlation matrix)
    b. NFACTORS= the number that actually exist (only with PA1 and PA2)
    c. MINEIGEN= 1.0 (only with PA1 and PA2)
    d. ITERATE= 25 (maximum) (only with PA2, RAO, ALPHA)
    e. STOPFACT= .001 (only with PA2, RAO, ALPHA)
7. FACSCORE allows for a raw data output of the final factor scores. If a variable has missing data, a factor score missing value is 999.0. "FACSCORE=" enables the user to specify the proportion of missing data that can be replaced by the mean and is a decimal number between 0 and 1.

Options -

1. Inclusion of missing data
2. Pairwise deletion of missing data
3. Input of correlation matrix
4. " " factor matrix and communalities
5. Output of simple correlation matrix
6. " " factor matrix and communalities
7. " " factor score coefficient matrix
8. " " means and standard deviations
9. Specifying the order of variables on input correlation matrices (only with Option 3)
10. Produce weighted factor scores when missing data is encountered
11. Sequence of output factor scores

Statistics -

1. Means and standard deviations
2. Correlation matrix
3. Inverse and determinant of correlation matrix
4. Communalities, eigenvalues, and proportion of total and command variance
5. Initial factor matrix
6. Rotated-factor matrix and transformation matrix
7. Factor score coefficient matrix
8. Plot of rotated factors

Further Information - Chapter 24 (pp. 468-509)

FREQUENCIES

Function - computes and presents one-way frequency distribution tables and optional descriptive statistics.

Applicable Variable Types - discrete or categorical (limited number of values)
- usually nominal or ordinal

Syntax -

```
1              16
FREQUENCIES    GENERAL= variable list
```

```
FREQUENCIES    INTEGER= var.(s) (min., max.), var.(s) (min., max.),...
```

Special Instructions -

1. FREQUENCIES operates under two modes:
    a. GENERAL (produces tables for any type of variable); and
    b. INTEGER ("   "   " only numeric integer variables).
2. A table will contain the values of the variable, the raw frequency of each category, the relative frequency distribution (figuring in missing values), the adjusted frequency distribution (excluding missing values) and a cumulative frequency distribution.
3. If a STATISTICS card is omitted, NO statistics will be printed.

Options -

1. Inclusion of missing values
2. Suppresses value labels
3. Prints the output in an 8½ X 11 inch space (for notebook binding)
4. Prints output on a permanent print file (tape or disk) (for this system, a FILE FILE9 card must be used: see Manual #3)
5. Condenses the table format
6. Condenses only tables that would normally require more than one page of output
7. Deletes the tables and prints only the requested descriptive statistics
8. Prints a histogram for each table
9. Prints a reference dictionary
10. Lists table with descending order of value (highest value first)
11. Lists table in descending order of frequency
12. Lists table in ascending order of frequency

Statistics -

1. Mean
2. Standard error
3. Median
4. Mode
5. Standard deviation
6. Variance
7. Kurtosis
8. Skewness
9. Range
10. Minimum
11. Maximum

Further Information - Chapter 14 (pp. 194-202)
Chapter 15 (pp. 216-217) in S.P.S.S. Update Manual

GUTTMAN SCALE

Function - performs scalogram analysis on any set of variables in the data. Provides a means of analyzing the underlying operating characteristics of 3 or more items in order to determine if their interrelationships meet the criteria for a Guttman scale (unidimensional, cumulative-ordered by degree of difficulty).

Applicable Variable Types - variables that are ordinal at least (or a dichotomous variable of the "yes-no" or "agree-disagree" nature)

Syntax -

```
1               16
GUTTMAN SCALE   scale name= var. (division point) var. (division point)...
                var. (division point)
```

Special Instructions -

1. A maximum of 12 variables may be entered on a single scale.
2. From one to three "division points" must be specified for each variable - cases that have values equal to or greater than the division point have "passed" the item. The "division point" may be whole or decimal.

Options -

1. Missing data entered into all scales and computations
2. Suppresses variable labels
3. Causes automatic ordering of variables from most to least difficult to be suppressed (if chosen, must enter the variables on the command in that order)

Statistics -

1. Inter-item and part-whole correlation coefficients
2. Coefficient of Reproductibility
3. Minimum Marginal Reproductibility
4. Percent Improvement achived by Guttman Scale
5. Coefficient of Scalability

Further Information - Chapter 26 (pp. 528-539)

MULT RESPONSE

Function - analyzes multiple response items instead of the use of many dichotomous items or content-coded variables. Analyzes simple variables, multiple dichotomies and multiple response items both individually and in conjunction with one another.

Applicable Variable Types - dichotomous, nominal

Syntax -

```
1               16
MULT RESPONSE   GROUPS- group name variable list (value list)/
                VARIABLES= variable list (min., max.)/
                FREQUENCIES= variables
                TABLES= variable list BY variable list BY.../
```

Special Instructions -

1. All four specifications may be made on one request, however, at least one of the first two must be present and one of the second two must be present to run this procedure.
2. On each specification, more than one sequence of "group name.../" or "variable list.../" may be made.
3. The GROUPS command identifies the multiple response items to be analyzed (multiple dichotomy groups or multiple content-coded variables). The group name must be unique and the variable list identifies the variables that make up the group.
4. The VARIABLES command identifies all the simple variables to be analyzed.
5. The FREQUENCIES command specifies the simple variables and groups for which frequency tables are desired.
6. The TABLES command specifies the n-way tables which are to be produced - must be the last statement in the procedure.

Options -

1. Inclusion of missing data
2. Groupwise deletion for multiple dichotomies
3. " " " " response groups (groups of content-coded variables)
4. Suppression of value labels
5. Base percentages on responses
6. Limit output width to 75 characters
11. Condensed frequency table format
12. " " " " for tables that would normally require more than one page

Statistics -

1. Inclusion of row percentages in the tables
2. " " columns " " " "
3. " " 2-way table total percentages in the tables

Further Information - Chapter 4 (pp. 99-109) in S.P.S.S. Update Manual

NONPAR CORR

Function - computes Spearman and/or Kendall rank-order correlation coefficients (nonparametric - does not depend upon a normal distribution). Requires the use of the variable's ranking, not absolute value.

Applicable Variable Types -

dependent: ordinal or higher
independent: ordinal or higher

Syntax -

```
1              16
NONPAR CORR    variable list WITH variable list
```

```
NONPAR CORR    variable list
```

Special Instructions -

1. The first illustration controls the variables to be correlated - only the variables before the 'WITH' will be correlated with the variables after the 'WITH'. The second syntax correlates every possible pair of variables.
2. If matrix output is desired, the second syntax must be used.
3. If no options are desired, only the Spearman correlation will be performed.

Options -

1. Inclusion of missing data
2. Listwise deletion of missing data
3. A two-tailed test of statistical significance for each coefficient
4. Writes a correlation matrix for future access
5. Only Kendall correlations are performed
6. Both Kendall and Spearman correlations are performed
7. If not enough workspace is available to handle all cases, a random sample is selected

Statistics -

none

Further Information - Chapter 18 (pp. 288-292)
Chapter 15 (pp. 221-222) in S.P.S.S. Update Manual

NPAR

Function - provides several nonparametric tests for analyzing variables for which no assumptions about the distribution are required.

Applicable Variable Types - depends upon the chosen test (listed below in brackets after each test name).

Syntax -

```
1               16
NPAR TESTS      test name = variable list/test name = variable list/...
```

Special Instructions -

1. The available tests are: (test names):
   - a. CHI-SQUARE (Chi-Square One-Sample Test) [nominal]
   - b. K-S (KOLMOGOROV-SMIRNOV One Sample and Two-Sample Test) [interval or higher]
   - c. RUNS (One-Sample Runs Test) [dichotomous]
   - d. MCNEMAR (McNemar Test for the Significance of Changes) [dichotomous]
   - e. SIGN (Sign Test) [ordinal]
   - f. WILCOXON (Wilcoxon Matched-Pairs Signed-Ranks Test) [ordinal]
   - g. COCHRAN (Cochran Q Test) [dichotomous, nominal]
   - h. FRIEDMAN (Friedman Two-Way ANOVA by Ranks) [ordinal]
   - i. MEDIAN (Median Test) [ordinal]
   - j. M-W (Mann-Whitney) [ordinal]
   - k. W-W (Wald-Wolfowitz) [ordinal]
   - l. MOSES (Moses Test of Extreme Reactions) [ordinal]
   - m. K-W (Kruskall-Wallis One-Way ANOVA by Ranks) [ordinal]
2. Each test has a unique syntax and statistics to go along with the test.

Options -

1. Inclusion of missing data
2. Listwise deletion of missing data
3. Special handling of variable lists
4. If too many cases to process, a random sample of cases is selected for analysis

Statistics -

1. Basic statistics (mean, maximum, minimum, standard deviation and number of cases)

Further Information - Chapter 3 (pp. 66-98) in S.P.S.S. Update Manual

ONEWAY

Function - performs a one-way analysis of variance - solves problems with only one independent variable.

Applicable Variable Types -

dependent: interval level
independent: nominal or ordinal

Syntax -

```
1              16
ONEWAY         dep. var.(s) BY indep. var. (min., max.)/
               POLYNOMIAL= #/
               CONTRAST= list of values/
               RANGES= test name (alpha)/
```

Special Instructions -

1. A maximum of 20 dependent variables may appear with one independent variable (of integer value).
2. The '(min., max.)' specifies the lowest and highest values to be used in the analysis.
3. Standard output of ONEWAY includes a summary table showing sums of squares, degrees of freedom, mean square, and the F-ratio (with its related significance).
4. Also available are tests for trends, a priori contrasts and posteriori contrasts:
    a. Tests for trends - "POLYNOMIAL= #/" partitions the between groups sums of squares into linear, quadratic, cubic or higher-order trend components. The '#' specifies the highest-degree polyninial to use - must be a positive integer less than or equal to 5 and less than the number of groups.
    b. A priori contrasts - "CONTRAST= list of coefficients to be assigned to various group means/" Up to 10 contrasts can be specified to be tested by the statistic 't'.
    c. A posteriori contrasts - "RANGES= test name (alpha)" tests pairs of group means. Up to 7 different tests may be requested. The tests are:
        1. LSD (Least Significant Difference) - any alpha
        2. DUNCAN (Duncan's Multiple Range Test)-.10, .05, .01 only
        3. SNK (Student-Newman-Keuls)- .05 only
        4. TUKEYB (Tukey's Alternate Procedure)- .05 only
        5. TUKEY (Honestly Significant Difference)- .05 only
        6. LSDMOD (Modified LSD)- any alpha
        7. SCHEFFE (Scheffe's Test)- any alpha

    The "(alpha)" is the specified significance level - if omitted, .05 will be used as a default.

Options -

1. Inclusion of missing data
2. Listwise deletion of missing data
3. Suppression of variable labels
4. Writes a number of cases, means, standard deviations for each group on a raw data output file
6. Uses the first 8 characters of the value labels of the independent variable as the group labels
7. Input is category (group) frequencies, means, and standard deviations not raw data
8. Input is category frequencies, means, pooled variance, and degrees of freedom for the pooled variance
10. Use harmonic mean for all groups in ranges test

Statistics -

1. Prints the number of cases for each group, means, standard deviations, standard errors, minimum, maximum and 95% confidence level for the mean
2. Prints both fixed- and random-effects measures
3. Prints homogeneity of variance statistics

Further Information - Chapter 22 (pp. 442-433)
Chapter 14 (pp. 209-212) in S.P.S.S. Update Manual

PARTIAL CORR

Function - computes partial correlation coefficients of any order or a combination of orders. Provides the user with a single measure of association describing the relationship between 2 variables while adjusting for the effects of one or more additional variables.

Applicable Variable Types -

dependent: interval level
independent: any level

Syntax -

```
1               16
PARTIAL CORR    correlation list of dep. and indep. variables BY list of control
                variables (order value(s))
```

Special Instructions -

1. Input for this procedure can be raw data, an S.P.S.S. system file or one or more simple correlation matrices.
2. The "correlation list" specifies the pairs of variables to be correlated. The dependent variable(s) is first, then the keyword WITH, then the independent variable(s).
3. The "control list" is a list of variables to be used as controls for each pair of variables specified in the correlation list.
4. "Order value(s)" specifies the exact partials which are to be computed. There can be a maximum of 5 values specified that can range from 1 to the number of variables in the two lists.
5. The results from this procedure are: partial correlation coefficients, the degrees of freedom, and a one- or two-tailed test of significance.

Options -

1. Inclusion of missing data
2. Pairwise deletion of missing data
3. Two-tailed tests of statistical significance
4. Matrix input
5. Writing matrices for future access
6. Specifying the order of variables for input matrices
7. Omission of degrees of freedom and significance
8. Printing of only non-redundant partials - in a serial string format

Statistics -

1. Prints out simple or zero order correlations used in computing the partial correlations
2. Means and standard deviations
3. Prints zero-order correlation coefficients if zero-order correlations are non-computable (99 will be printed).

Further Information - Chapter 19 (pp. 301-319)

PEARSON CORR

Function - performs bivariate correlation analysis by calculating Pearson product-moment correlation coefficients for pairs of variables indicating the degree to which variation in one variable is related to variation in the other.

Applicable Variable Types -

dependent: interval
Independent: interval

Syntax -

```
1              16
PEARSON CORR   dep. var.(s) WITH indep. var.(s)
```

```
1              16
PEARSON CORR   variable list
```

Special Instructions -

1. The correlations performed are zero-order correlations where the Pearson r coefficient measures the strength of the relationship between the two variables and r^2 measures the proportion of variance in one variable explained by the other.
2. Standard output will give 'r', the observed significance level and the number of cases upon which the coefficient is based.
3. The first illustration allows user control of the pairs to be correlated - computes a correlation coefficient for each variable before the WITH with each variable after the WITH.
4. The second illustration creates coefficients for every possible combination of variables in the list (must use this syntax if matrix output is desired).

Options -

1. Inclusion of missing data
2. Listwise deletion of missing data
3. Two-tailed test of significance
4. Writes a correlation matrix as saved output
5. Omits number of cases and significance levels
6. Prints only non-redundant coefficient in a serial string format

Statistics -

1. Means and standard deviations
2. Cross-product deviations and covariance

Further Information - Chapter 18 (pp. 276-288)

REGRESSION

Function - combines standard multiple regression and stepwise procedures thereby providing considerable control over the inclusion of independent variables in the regression equation. Analyzes the relationship between a dependent variable and a set of independent (or predictor) variables.

Applicable Variable Types -

dependent: interval or higher
independent: interval or higher (although use of nominal variables as "dummies" is allowed)

Syntax -

```
1              16
REGRESSION     VARIABLES= variable list/
               REGRESSION= dep. var. WITH indep. var.(s) (#)
```

Special Instructions -

1. Input for this procedure can be raw data, an S.P.S.S. system file or a correlation matrix.
2. Several "REGRESSION=" statements are allowed on one Regression request and variable list and are separated by a slash.
3. Only the variables defined the "VARIABLES=" statement may be used in any associated regression, however, not all have to be used.
4. An initial correlation matrix is formed from the "VARIABLES=" statement from which the regression(s) is computed.
5. The "REGRESSION=" statement specifies the dependent variable and which variables are to be used as predictors.
6. For a simple regression, the number in parenthesis must be an even number between 2 and 98.
7. For a hierarchical regression, each independent variable (or block of independent variables) must be followed by a different inclusion level (number in parenthesis). Variables with higher inclusion levels are entered into the equation before ones with lower numbers (e.g. VAR1, (8), VAR2 (2), VAR3 (6), VAR4 (4)/ VAR1 will be entered first, then a regression with VAR1 and VAR3 will be done, then a regression with VAR1, VAR3 and VAR4 will be done, and, lastly, a regression with VAR1, VAR3, VAR4, and VAR2 will be performed).
8. For a regression that allows stepwise inclusion, independent variables are entered only if they meet certain statistical criteria. The order of inclusion is determined by the respective contribution of each variable to explained variance. The number in parenthesis must be an odd number from 1 to 99.
9. Stepwise inclusion can be performed in conjunction with hierarchical inclusion by following each independent variable (or block of independent variables) with a different odd number (the highest one(s) entered first).

10. In a stepwise regression, the parameters for the decision to include a variable default at (80, .01, .001) or (n, f, t) - the first number is the maximum number of independent variables entered, the second number is the F ratio of a regression coefficient not yet in the equation and the third number is the tolerance of an independent variables (the proportion of variance in the variable to be entered not explained by the present independent variables.
11. If residuals and predicted 'Y' values are desired, the user must insert a 'RESID=mdrp' statement after the last inclusion level number in a regression request. The 'mdrp' stands for missing data replacement proportion and is a numeric value between 0 and 1.

Options -

1. Inclusion of missing data
2. Pairwise deletion of missing data
3. Suppression of labels
4. Matrix input
5. Input of means and standard deviations
6. Suppression of step-by-step output
7. Suppression of summary table
8. Matrix output
9. Input correlation matrix is indexed by VARIABLES= statement
10. Causes sequencing information to be entered in columns 1 through 20 of each record in the raw data output file
11. Standardized residuals on output file
12. Y' values on output file
13. Creates standardized predictors which are a weighted product of existing data (only with Option 2 and use of RESID=mdrp of 0)
14. Suppresses axes on the plots of standardized predictor versus standardized residual
15. Means and standard deviations are output

Statistics -

1. Prints correlation matrix
2. Means, standard deviations, and number of valid cases
3. Forced printing of correlation matrix and warning of bad elements
4. Output of a plot of Standardized residuals against the sequence of cases in a file
5. Computes Durbin-Watson statistic for residuals
6. Requests a plot of standardized residuals against standardized Y' values with residuals on the vertical axis
7. Printout of correlation matrix and number of cases

Further Information - Chapter 20 (pp. 320-367)
Chapter 15 (p. 223) in S.P.S.S. Update Manual

RELIABILITY

Function - evaluates multiple-item additive scales through the computation coefficients of reliability.

Applicable Variable Types - ordinal

Syntax -

```
1              16
RELIABILITY    VARIABLES= variable list/
               SCALE= (scale label) scale list/
               MODEL= keyword/
```

Special Instructions -

1. The VARIABLES command assembles a triangular covariance matrix and vector of means.
2. The SCALES command selects (from the 'VARIABLES=') the variables that are to be part of a particular scale. More than one of these statements may be specified and at least one must be specified.
3. The MODEL command controls the computations to be performed and how the results are printed in conjunction with the STATISTICS and OPTIONS statements. The keywords are:

   ALPHA (Cronbach's Alpha and Standardized Item Alpha)
   SPLIT (Split-Half Coefficients)
   GUTTMAN (Guttman's Lower Bounds for True Reliability)
   STRICTPARALLEL (Maximum Likelihood Reliability Estimate)
4. The input for this program can be raw data, an S.P.S.S. system file, correlation matrix or a covariance matrix.

Options -

1. Inclusion of missing data
3. Suppression of variable labels
4. Input is a covariance matrix
5. "   " " vector of standard deviations and a correlation matrix
6. "   matrix is triangular
7. A vector of means precedes the matrix
8. Writes a covariance matrix as output
9. Index matrices using S.P.S.S. variable list
10. Matrices and vectors output without performing scale analysis
11. Writes means as output
12. Input matrix is written as a vector
13. Write matrix as a single vector
14. Solution method Z
15. Freidman's Chi-square
16. Cochran's Q

Statistics -

1. Item means and standard deviations
2. Inter-item variance co-variance matrix
3. " " correlations
4. Scale means and scale variance
5. Summary statistics for item means
6. " " " " variances
7. " " " inter-item covariances
8. " " " " " correlations
9. Item total statistics
10. Analysis of Variance table
11. Tukey test for Additivity
12. Hotelling's T-Squared

Further Information - Chapter 5 (pp. 110-137) in S.P.S.S. Update Manual

REPORT

Function - produces case listings and aggregated summary statistics in an easily understood format for people unfamiliar with typical data calculations and output (e.g., businesses, schools, hospitals, organizational groups).

Applicable Variable Types - any

Syntax -

```
1              16
REPORT         FORMAT= keyword (#), keyword (#),,.../
               VARIABLES= list of variables/
               BREAK= one variable name/
               SUMMARY= list of keywords/
```

Special Instructions -

1. The FORMAT command establishes the physical dimensions of the printed output and the vertical spacing around headings. Any or all of the following may be specified:
   a. HDSPACE (#)- the number of lines between the report head and column heads (default=3)
   b. CHDSPACE (#)- the lines of space beneath the longest column head (default=1)
   c. BRKSPACE (#)- the lines of space beneath the break head (default=1)
   d. FTSPACE= lines of space between the last listing and the running foot (default=1)
   e. LENGTH (#,#)- number of lines to be dropped from the top or raised from the bottom to center the page (default for both=0)
   f. MARGINS (#,#)- left and right margins (defaults=1, 132)
   g. LIST-requests summary statistics be printed across breaks as well as for each break (default=NOTOTAL)

   If all the defaults are desired, use "FORMAT=DEFAULT/".
2. The VARIABLES command identifies the variables to be included in the report and defines the output columns for each variable. It is possible to incorporate column headings by this syntax: varname '10 character head' '10 character head'..., varname '10 character head' '....' (each up to 10 character head resides on one line, the following associated head appears beneath the first and so on).
3. The BREAK command specifies the variable or set of variables which define a subpopulation within the report. It may be requested more than once on the same report. It defines the break level and the format of the break head and must be requested at least once and directly followed by a SUMMARY command that specifies the desired statistics. If secondary breaks are desired, another BREAK and SUMMARY command sequence should follow. The user can specify a break title in single quotes after the variable name (default= the variable name), the width of the column (default= 10 spaces), and whether a new page should be started for each new break (default= 2 line skip).

4. The SUMMARY command specifies the aggregate statistics to be computed for the variables on the 'VARIABLE=' statement for each break. Every break has a summary - but if, for lower level breaks, a SUMMARY is not specified, the system will assume the summary for the first (primary) break. The following are the available aggregared statistics:

   VALIDN
   VARIANCE
   SUM
   MEAN
   STDEV
   MIN
   MAX
   SKEWNESS
   KURTOSIS
   PCTGT (#)
   PCTLT (#)
   PCTBTN (#,#)
   ABFREQ (min., max.)
   RELFREQ (min., max.)
   MEDIAN (min., max.)
   MODE (min., max.)

   Composite functions are also available:

```
DIVIDE (agg. stat. (var. name) agg. stat. (var. name)) divides first
by second
PCT (agg. stat. (var. name) agg. stat. (var. name))
SUBTRACT (agg. stat. (var. name) agg. stat. (var. name))
ADD (agg. stat. (var. name)...agg. stat. (var. name))
GREAT (    "         "          "           "    ))
LEAST(     "         "          "           "    ))
AVERAGE(   "         "          "           "    ))
```

5. There are mechanisms for dealing with variable names of larger than 8 characters (STRING=) and dealing with missing data (MISSING=) and many other report format specifications to override defaults for column size, etc.

<u>Options</u> -

none

<u>Statistics</u> -

none

<u>Further Information</u> - Chapter 1 (pp. 1-37) of S.P.S.S. Update Manual

SCATTERGRAM

Function - prints two-variable scatter diagrams of data points - A graphical representation of the relationship between two variables where the coordinates are the values of each variable.

Applicable Variable Types -

dependent: interval or higher
independent: interval or higher

Syntax -

```
1               16
SCATTERGRAM     variable list
```

```
1               16
SCATTERGRAM     variable list WITH variable list
```

```
1               16
SCATTERGRAM     varialbe list (low, high), variable list (low, high) WITH
                variable list (low, high), variable list (low, high)...
```

Special Instructions -

1. The first syntax will result in a graph of every possible combination of variables with each variable assuming both axes.
2. The second syntax enables user control over the graphs to be printed. The variables before the 'WITH' will assume the vertical axis for each variable after the 'WITH' (horizontal axis).
3. The third syntax enables user specification of the lowest and highest value for any or each axis in any or all graphs. If no specification is made, the axes on the graph will begin and end with the original values for each variable.

Options -

1. Inclusion of missing data
2. Listwise deletion of missing data
3. Suppression of variable labels
4. Grid lines on the plot are suppressed
5. Prints diagonal grids
6. Two-tailed test of significance (with statistic 3)
7. Automatic scaling (integer plot labels)
8. If not enough space is available, a plot will be printed for a random sample of cases in the file

Statistics -

1. Pearson's r
2. r^2
3. Significance of r
4. Standard error of the estimate
5. Intercept with vertical axis
6. Slope

Further Information - Chapter 18 (pp. 293-300)

SURVIVAL

Function - Evaluates the time between two events (a starting event and terminal event) - used to describe and compare the time intervals for different groups.

Applicable Variable Types -

survival variables - interval (time);
status and controlling variables - any level

Syntax -

```
1             16
SURVIVAL      TABLES= survival variable list BY control variable list
                      (lo, hi) BY.../
              INTERVALS= THRU # BY #, THRU # BY #.../
              STATUS= ststus variable (lo, hi) FOR survival variable list/
              PLOTS (keyword)= survival variable list BY variable list BY.../
              COMPARE= survival variable list BY variable list BY.../
```

Special Instructions -

1. The TABLES command establishes the basic configuration of the survival analysis by listing the dependent (survival) variables followed optionally by control variables with their range in parenthesis (maximum of 2 sets of controls).
2. The INTERVALS command establishes the length of the period under examination and the intervals into which it should be broken. The first number is the number of units of time to be examined (e.g. 120 days/weeks - however the data is recorded) and the second number is the number of intervals into which this span should be broken (2 or 4 or 3, etc.).
3. The STATUS command associates a status with each survival variable. More than one 'STATUS=' statement may appear if different survival variables have different status specifications.
4. The PLOTS command requests line printer plots of the survival functions against time. If omitted, plots will be produced for all variables in the procedure - same as PLOTS (ALL) syntax. Other plot selection keywords:

   PLOTS (LOGSURV)=______________________/
   PLOTS (SURVIVAL)=_____________________/
   PLOTS (HAZARD)=_______________________/
   PLOTS (DENSITY)=______________________/

   More than one keyword can be contained in the parenthesis.
5. The COMPARE command requests comparison of the survival of subgroups in the sample (control variable values). If omitted, the list on the 'TABLES=' statement will be used.

Options -

1. Inclusion of missing data
2. Casewise deletion of missing data
3. Computes comparisons only
4. Suppresses labels
5. Default to approximate comparisons
6. Do approximate comparisons
7. Perform pairwise comparisons
8. Output Survival Table Data Records
9. "   "   "   "   and label records

Statistics -

none

Further Information - Chapter 2 (pp. 38-65) in S.P.S.S. Update Manual

T-TEST

Function - tests the significance of the difference in means of a variable in two independent groups or of two variables in the same set of cases. Computes Student's t and probability levels to discover differences in subgroups or in effects of variables in a group.

Applicable Variable Types - ordinal or interval (variables for which a mean is a reasonable measure of central tendency) - for groups, any level of measurement.

Syntax -

```
1              16
T-TEST         GROUPS= group specification/VARIABLES= variable list
```

```
T-TEST         PAIRS= variable list
```

```
T-TEST         PAIRS= variable list WITH variable list
```

Special Instructions -

1. The first syntax is for an Independent Samples T-Test (between groups). The "group specification" may be of several types:

   a. If Group 1 is formed of all cases where a variable is *greater than or equal to* a certain value and Group 2 is all remaining cases, the group specification would be:

   ```
   GROUPS= var. name (value)/
   ```

   b. If Group 1 is formed by all cases where a variable equals value 1 and Group 2 is all cases where the variable equals value 2, the statement would look like:

   ```
   GROUPS= var. name (value 1, value 2)/
   ```

   c. If the variable that determines the 2 groups has only 2 values:

   ```
   GROUPS= var. name/
   ```

   d. If the groups are formed by taking the first n cases for Group 1 and the next n cases for group 2:

   GROUPS= n_1, n_2/

2. The variable list in each syntax is the variables for which means are to be compared and tested.
3. The second and third syntaxes illustrate, Paired Sample T-Tests (or between variables). The first of these will result in every possible combination of two variables in the list to be tested. The second of these allows user control over what variables are to be compared (only the ones before the 'WITH' will be compared with each variable after the 'WITH').
4. Both types of T-Tests can be requested on one procedure request - the GROUPS must come first then the PAIRS (separated by a slash).

Options -

1. Inclusion of missing data
2. Listwise deletion of missing data
3. Suppression of labels

Statistics -

none

Further Information - Chapter 17 (pp. 267-275)

WRITE CASES

Function - writes all or parts of data (original and/or modified or created) on a chosen medium (usually disk or tape) thereby creating a new raw data set (file).

Applicable Variable Types - any

Syntax -

```
1              16
WRITE CASES    (fortran statement) variable list
```

Special Instructions -

1. Variables on this procedure must be previously defined but need not follow any specific order.
2. Variables may be entered more than once.
3. The format statement specifies the location, number and size of the variables to be written - a set of instructions for writing the data.
   a. Only alphanumeric (A-format) and numeric (F-format) variables may be written.
   b. A-type variables can only be 4 columns wide (A4); F-type variables can be only 9 columns wide (F9.0, F9.1, F9.2,..., F9.9).
   c. A slash (/) is used to denote the end of a card (or record) and may be used to skip entire cards.
   d. An 'X' is used to denote skipping a space.
   e. To have something standard written on a record, put it in single quotes (e.g., (F4.0, 2X, 'CARD 1', 3F7.0...) will write "CARD 1" on that record for each case).
   f. For multiple variables with the same specification, put the number of duplicates before the string (e.g., 10(F3.0, 1X, A3) will write that particular format 10 consecutive times).
   g. Decimals are not printed on the created data record, so, when figuring how much space to allocate, do not include the decimal as a column in the width of a variable.
   h. A negative sign must be included in the total field width.
   i. All numbers are right justified so if a variable is smaller than the specified field, blanks will fill up the left of the value.
   j. If a field is too small for a variable, asterisks will be printed in the field.
4. Make sure that each variable in the variable list has a corresponding format.
5. The end of one card and beginning of another is not denoted in the variable list - just continue with the variable names as they appear on the new data.
6. Only one format list and corresponding variable list may be requested on a single WRITE CASES procedure.

Options -

1. Any case with missing data on any variable on the WRITE CASE will not be output

Statistics -

none

Further Information - Chapter 10 (pp. 139-144)

APPENDIX 1

CHART OF ARITHMETIC OPERATORS AND MATHEMATICAL FUNCTIONS

Arithmetic Operators

SYMBOL	MEANING	EXAMPLE
/	Division	VARX = VARA/VARB
*	Multiplication	VARX = VARA*VARB
+	Addition	VARX = VARB
-	Subtraction	VARX = VARA-VARB
**	Exponentation	VARX = VARA**2

Mathematical Functions

SYMBOL	MEANING	EXAMPLE
SORT	Square Root	VARX = SORT (VARA)
LN	Natural logarithm	VARX = LN (VARA)
LG10	Base 10 logarithm	VARX = LG10 (VARA)
EXP	Exponential (e^{arg})	VARX = EXP (VARA + VARB)
SIN	Sine	VARX = SIN (VARA + VARB)
COS	Cosine	VARX = COS (VARA)
ATAN	Arctangent	VARX = ATAN (VARA)
RND	Round result to whole number	VARX = RND (VARA + VARB/6)
ABS	Absolute number	VARX = ABS (VARA - VARB)
TRUNC	Truncate value (whole number w/o rounding)	VARX = TRUNC (VARA/7)
MOD10	Result is a remainder of division by 10	VARX = MOD10 (VARA)

APPENDIX 2

RELATIONAL OPERATORS

Relation	Keyword	Definition
Greater than or equal to	GE	If expression 1 is greater than or equal to expression 2, then the value of the relation is true; otherwise it is false.
Less than or equal to	LE	If expression 1 is less than or equal to expression 2, the value of the relation is true.
Greater than	GT	If expression 1 is greater than expression 2, the value of the relation is true.
Less than	LT	If expression 1 is less than expression 2, the value of the relation is true.
Equal to	EQ	If expression 1 is exactly equal to expression 2, the value of the relation is true.
Not equal to	NE	If expression 1 is not exactly equal to expression 2, the value of the relation is true.

LOGICAL OPERATORS

Keyword	Definition
AND	The resulting logical expression will have value true if and only if the relations directly preceding and following the operator have the value true.
OR	The resulting logical expression will be true if either or both of the constituent relations has the value true.

APPENDIX 3

Reference List

S.P.S.S. References

Statistical Package for the Social Sciences, Second Edition, Norman H. Nie, et. al., New York: McGraw-Hill Book Company, 1975.

S.P.S.S. Update: New Procedures and Facilities for Releases 7 and 8, C. Hadlai Hull and Norman H. Nie, New York: McGraw-Hill Book Company, 1979.

S.P.S.S. Primer, William R. Klecka, et. al., New York: McGraw-Hill Book Company, 1975.

S.P.S.S. Introductory Guide: Basic Statistics and Operations, Marija J. Norusis, New York: McGraw-Hill Book Company, 1982.

A Data Analysis Handbook Using the S.P.S.S. System, Michael J. Cleary, Ph.D. and Robert T. Amsden, Ph.D., Ohio: Advocate Publishing Group, 1980.

S.P.S.S. USER'S MANUAL # 3:

SYSTEM CONTROL LANGUAGE AND S.P.S.S.

Prepared by

Lorna L. Earl

TABLE OF CONTENTS

Introduction.. 1

Standard Control Statements For Any Run (BEGIN JOB, USER, RUN SPSS, DATA IOCR, END JOB)............................... 1

Running S.P.S.S. Using The Disk

- Creating A Job File 4
- Running The Program From The Terminal....................................... 6
- Getting A Program From The Library Into The Workspace........................ 7
- Removing A Program From The Workspace And/Or Library........................ 7
- Finding Out What Programs Reside In The Library............................. 7

Factors Which Influence The Inclusion Of System Control Statements Within The Program

The Form in Which The Data Exist

1. Raw Data That Exist Within The Program.................................. 8
2. Raw Data That Exist On Magnetic Tape..................................... 10
3. Raw Data That Is Stored On Disk.. 11

The Status Of The File For The Run

1. Temporary Files.. 12
2. Creating A Permanent S.P.S.S. System File................................ 12
3. Accessing A Permanent S.P.S.S. System File............................... 13
4. Manipulating Permanent Files (Merging All Or Parts of Various Files)... 14

The Method of Handling Output

1. Printed Output... 16
2. Raw Data Output.. 17

Running S.P.S.S. On Cards... 18

Appendices

Appendix 1

Location of Terminals... 20

Appendix 2

Basic Operating Instructions For The Hard Copy And CRT Terminals........... 21

Appendix 3

Basic CANDE Commands.. 24

Appendix 4

Reference List.. 30

INTRODUCTION

This manual is written for the "advanced" S.P.S.S. user. Because this pre-supposes a familiarity with program construction and syntax, this manual will deal only with using S.P.S.S. on the P.S.U.C. system (Burroughs B6800). For more detailed documentation of S.P.S.S., see the other manuals in this series.

The materials presented in this manual are intended to supply the user with the basic information needed to run S.P.S.S. on this system. The system, however, has many facilities that can be used to run many types of programs. For a comprehensive presentation of WFL and CANDE commands*, reference manuals are available (see Appendix 4).

Although similar control statement preparation is required for each run of S.P.S.S., the inclusion of certain system control statements (file equate statements) depends upon four factors. They are:

1. The form in which the program exists (on cards for Batch submittal or stored as a job file on disk)
2. The form in which the data exist (card, tape or disk)
3. The status of the file for the run (temporary or permanent) and the nature of file manipulation (if desired)
4. The method of handling output (printer, remote, storage on tape or disk)

STANDARD CONTROL STATEMENTS FOR ANY RUN

To use S.P.S.S. on this system, the following statements must be present in each program to be run:

The "BEGIN JOB" statement

This is the first statement in each program. In the first column, an illegal character must appear (either in multi-punch on cards or a '?' for job files on disk). The format is:

```
1
?BEGIN JOB__job name__;CHARGE = 5 digit number;
⅓BEGIN JOB__job name__;CHARGE = 5 digit number;
```

The job name can range from 1 to 17 characters and is printed on the first page of your printed output. It serves to identify the S.P.S.S. job.

---

*These acronyms stand for Work Flow Language and Command AND Edit language respectively. See Appendix 3.

The 5 digit number is the same number as the user's departmental chargecode or account number. The exact number must appear otherwise, access to the system will not be allowed. To find out or verify the correct departmental chargecode, contact the Conputer Center (564-3013: 5th floor of the Administration Building).

The "USER" statement*

This statement must appear in order to identify the user to the system and verify legal access. The usercode consists of the first 6 letters of the user's last name (if the name is less than six letters, continue using the letters of the first name), a 3 letter departmental code and a password which serves to protect the user's file.

```
_ _ _ _ _ _ _ _ _ _/ _________________________
6 letters   depart-    password (not to exceed
of last     mental     17 characters)
name        code
```

-When first issued, the password associated with the usercode is the word 'FACULTY' if the user is a faculty member and the word 'STUDENT' if the user is a student. It is recommended that this default password by changed to one that is unique to insure protection of files. A measure for keeping the unique password secret can be done when typing it on cards, put the password on a separate card and flip off the 'PRINT' switch; on the terminal, press the ESC key, then the 'G' then the '1' which will erase each character typed.

-The mechanism for changing the password is:

a. in a card deck, place this statement directly after the "USER" card:

```
1
PASSWORD=new password;
```

b. when using a terminal, type in:

```
1
PASSWORD
```

after you are logged on (can be changed at any time). Follow the directions: on the messages: enter the present password, the new password, and re-enter the new password.

---

*To obtain a usercode, fill out the appropriate form which is available at the Computer Center.

The "RUN SPSS" statement

Without this statement, the computer will not recognize the program as S.P.S.S. and will, therefore, not summon the appropriate software. The card appears directly after the "USER" statement in a card deck, or the "BEGIN JOB" statement in a job created on disk and before any file equate statements. The general format is:

```
1
RUN SPSS;
```

The "DATA IOCR" statement

This statement signals the system that the following program will require some form of data to be read (analyzed) before the program (or job) ends. If not inserted, S.P.S.S. will not recognize the data as data and treat each card (or record) as an S.P.S.S. statement. In addition, this statement signals the system that the statements to follow are part of the S.P.S.S. program. This appears directly before your first S.P.S.S. statement and after any or all file equate statements. The general format is:

```
1
DATA IOCR
```

The "END JOB" statement

This is the last statement in the program (or job) signaling the system that the program is finished. As with the first statement in the program, the first column carries on illegal character (multi-punch on cards or '?' for a disk-created job). The format is:

```
1
?END JOB
   OR
▮END JOB
```

These five system control statements are mandatory for every S.P.S.S. run. The other various control statements available for inclusion are contingent upon the form and purpose of the program. The remainder of the manual, then, will present each contingency and the proper control commands for a variety of alternatives which lie therein. This section will contain examples of running S.P.S.S. on terminals (using jobs stored on disk) to illustrate the appropriate control statements. Information on using card decks for "Batch" jobs will be discussed later with illustrations only for those features which are different when using cards as opposed to the terminal to run S.P.S.S.

RUNNING S.P.S.S. USING THE DISK

Creating a Job File

S.P.S.S. is run on disk through a job file that is created, saved and activated from the terminal.* This job file houses all control statements and S.P.S.S. commands to completely run a program.

After logging onto the system, type this statement:

```
MAKE  job name JOB
```

The job name labels a job file and it is this title that will appear when a listing of files is requested. It can contain up to 17 characters (letters and/or numbers but not embedded blanks or special characters) and up to a series of 11 names for one job (each separated with a slash). This name can be the same as the one on the BEGIN JOB statement of the program.

The computer will respond with a verification that a job file has been created with that name.

Because S.P.S.S. mandates that the specification field begins in column 16, it is recommended that a tab is set that will automatically (whenever it appears) begin all that is after it in column 16. Type this statement:

```
TAB=_ 16
```

The underline signifies the choice of one special character that will represent the tabulator (e.g. % or $). This character can be any unique symbol that will not be used elsewhere (otherwise, every time this non-unique symbol appears, all else following will start in column 16 or every 16th column after that). After typing in the TAB=character, skip one space and type 16.

The computer will respond with a # to signify that the tab has been set.

This system requires that every line in a job file be preceded with a sequence number. It is recommended that an increment of a large enough size be assigned (e.g. 10, 20, 50, or 100) to allow for future insertion of new lines between the pre-existing ones. Sequence numbers can be entered in one of 2 ways: the user can type in the numbers or the system will automatically supply each line number. To let the system assign sequence numbers, type this statement:

---

*Faculty and staff have priority access to several hard-copy terminals interspersed throught the campus. See Appendix 1 for the location of these terminals and Appendix 2 for basic operation instructions.

```
SEQ
```

With this statement, the first line number will be 100, the next, 200, and so on because the default on sequence number increments is 100.

```
SEQ 1+10
```

This statement will start the line numbers at 1 and increment each line number by 10.

```
SEQ 50+20
```

This statement will assign the first line number at 50 and increment by 20.

The general syntax is:

```
SEQ base+increment
```

The computer will respond with the first line number and, every time the RETURN key is hit, the next line number will appear. Do not start typing a line until the number appears. (Note: If response time is slow, the numbers will appear slowly. At such times, it is more time-efficient for the user to type in the sequence numbers.) NOTE: While in the sequence mode, to make the illegal character on the first and last statement of the program, type the '?' TWICE.

When finished typing the program, get out of the sequence mode by hitting the RETURN key twice. Remember, nothing else can be done until out of the sequence mode (otherwise, every line typed in becomes part of that program or job).

```
   program statements
12000FINISH [RETURN]
12100??END JOB[RETURN]
12200[RETURN]
```

When out of the sequence mode, a number of operations can be done on the job file. A listing of what was typed, corrections, line changes, and so on are possible until the job file houses the desired S.P.S.S. program.(For simple editing commands through CANDE, see Appendix 3).

The final step in making a job file to run S.P.S.S. is saving the job. Until now, the job was created and manipulated in a temporary workspace, something that will disappear when the user logs off the system. To permanently save the job in the user's library, type this command after all editing has been completed:

```
SAVE
```

The computer will respond with a verification that a job file with this name has been saved.

NOTE: If this command is forgotten and the user tries to sign off, the computer will "remind" the user by printing:

```
REMOVE OR SAVE WORKFILE
```

NOTE: Once saved, each time the user works with a job file, 2 copies exist: one in the workspace, subject to changes by the user, and one in the Library which remains in tact until the next SAVE command.

Running the Program from the Terminal

After a job file has been saved, it can be run.* To run the job file that will process the S.P.S.S. program type this statement:

```
START job name
```

The computer will respond with a series of messages:

#RUNNING (number) [the number of the job that is presently being run]

#JOB (number) in QUEUE (number) [the number and queue of the user's job to be run]

[denotes that when the preceding job(s) are finished, the user's job will be run automatically - when this character appears, it is safe to sign off or get or start another job - the output will not go to the printer until the user signs off]

*It is possible to manipulate a job file in the temporary workspace and start this updated job without altering the saved library job file. At the end of the run the temporarily updated job will produce output and can be removed thereby leaving the original job file in tact. In this case, just type:

```
START
```

The computer will process the updated job file in the temporary workspace; but, upon logging off the system, the changes made will be "forgotten."

# (number) BOJ (job name) [denotes that the job has started]

(num)/(num) BOT *SPSS on ACADEMIC [denotes that S.P.S.S. has been summoned]

# (num)

wait while program is running

# (num)/(num) EOT SPSS on ACADEMIC [denotes that S.P.S.S. is finished]

# (num)/(num) EOJ JOB (job name) [denotes that the job is finished]

Getting a Program from the Library into the Workspace

To work with a program that is not in the workspace (because either the user has just logged on or another program presently exists in the workspace), call the job that houses it by typing:

```
GET job name
```

The computer will respond by reporting the job name and the number of records (lines) that comprise the job file. That job file is now in the workspace to look at, edit, re-save, (if edited), and/or start.*

Removing a Program from the Workspace and/or Library

If a job file is no longer needed, it may be removed from the library permanently by typing:

```
REMOVE job name
```

To permanently remove the job file that resides in the workspace at the present time, type REMOVE then REMOVE job name. The first REMOVE removes the job from the workspace and the second permanently removes the job from the library.

Finding Out What Programs Reside in the Library

To get a listing of all the files (and their types) that reside in the library, type:

```
FILES
```

The computer will respond with a heading of the user's usercode, the pack on which the files are stored and the names and kind of all files that have been saved.

---

*It is not necessary to bring a file into the workspace to look at it or activate it. Just type:

```
LIST  job name
START job name
```

However, to edit and re-save a job file, it must be summoned into the workspace by using the GET command.

FACTORS WHICH INFLUENCE THE INCLUSION OF SYSTEM CONTROL STATEMENTS WITHIN THE PROGRAM

The Form in Which the Data Exist

Because the raw data may exist on one of several mediums, the correct file equate statement to locate and define the data for the system must be used.* The appropriate file equate statements and example program constructions will be presented for data that appears on the following mediums: 1. within the program; 2. on magnetic tape; 3. on disk file.

1. Raw Data that Exist Within the Program

If the raw data is physically present in the program, no file equate statment is needed. A job file that houses an S.P.S.S. program that contains raw data can exist by two mechanisms. A pre-existing card and data deck can be read directly and stored in a job file on disk by use of this statement as the first card in the deck to be entered:

```
BEGIN JOB______ WITH NEWSOURCE= name of job file_______________; CHARGE=_____;
USER=_______________/_________________________;
RUN SPSS;
DATA IOCR
 |
SPSS program
 |
data cards
 |
END JOB
```

After this program has been run, the entire deck will be stored in a job file with the specified name to manipulate from the terminal. To merely read in a program and data deck without actually running S.P.S.S., type this first card:

```
BEGIN JOB______________________ NEWSOURCE SYNTAX; CHARGE=_______________;
```

This run will result nothing but storing the deck in a job file under the job name on the first card.

*IMPORTANT NOTE: Specific numbers have been reserved for specific functions of file-equate statements. The number (e.g. FILE3, FILE15) presented is the only one that can be used for the particular condition being discussed.

When using NEWSOURCE, the first and last illegal punch will be replaced by a '?'. It is important to note, however, that the first statement must be altered by removing the 'WITH NEWSOURCE' or 'NEWSOURCE SYNTAX' from the statement. It is also recommended that the 'CHARGE=________;' and 'USER=_________;' statements be removed as they are redundant pieces of information for the execution of the job.

The second mechanism that allows data to exist within the program is typing the program and data records directly into the terminal in a job file. If the sequence mode is used, wait for the sequence number to appear and type the data records. For data that begins in the first column, do not skip a space after the sequence number; if a space is skipped, the data will be placed in column 2. In the sequence mode, this data record on line 1500 with no space after the sequence number:

```
150021 3 8 5
```

will appear as

```
1500 21 3 8 5
```

when listed. Correspondingly, this data record typed in with a space after the sequence number:

```
1600 31 2 7 4
```

will appear as:

```
1600  31 2 7 4
```

when listed with the '31' beginning in column 2.

NOTE: If the user does not employ the sequence mode when entering the program and data, the data must begin in at least column 2. The computer will not recognize the end of the sequence number and the beginning of the data record if no space is skipped. The following lines entered without the sequence mode:

```
2000READ INPUT DATA
210003 50 40
220045 67 90
2300FINISH
2400?END JOB
```

will appear as

```
2000 READ INPUT DATA
2300 FINISH
2400 ?END JOB
210003 50 40
220045 67 90
```

When entering data as well as the program, then, it is wise to use the sequence mode to avoid such idiosyncratic problems.

2. Raw Data that Exist on Magnetic Tape

For data that resides on a medium other than physically present in the program, a file statement is required to locate the data for the system. The general form of this statement is:

```
1
FILE FILE8 (KIND=medium, TITLE=__________);
```

This statement appears after the RUN SPSS and before the DATA IOCR statements. The medium is usually either TAPE or DISK and the title is the label or name of the data file. For data on a tape, the program set-up would be as follows:

```
1
?BEGIN JOB________________________;
RUN SPSS;
FILE FILE8 (KIND=TAPE,TITLE=MYTAPE);
DATA IOCR

  S.P.S.S. program

INPUT MEDIUM TAPE

READ INPUT DATA

FINISH
?END JOB
```

This file statement contains the minimum information the system needs to find and read the raw data. Before a tape can be accessed, however, several matters must be addressed. The tape must be physically mounted by a computer operator. This requires that the computer center is made aware that the raw data tape will be required for a run. The number of the tape*, the title or the tape** and the approximate time of the run are all necessary information about which the Computer Center must be made aware when using data stored on a tape.

3. Raw Data that is Stored on Disk

Data records can be stored in a disk file and accessed by the same file equate statement.

```
1
?BEGIN JOB____________________;
RUN SPSS;
FILE FILE8 (KIND=DISK,TITLE=DISKFILE);
DATA IOCR

 S.P.S.S. program

INPUT MEDIUM DISC

READ INPUT DATA

FINISH
?END JOB
```

The advantage of having data in a disk file is that immediate access is available without having to contact the Computer Center. It is important to remember, however, that prudent use of disk space is paramount to insure equal distribution of this limited resource. It is possible, however, to store small data sets in disk files or store large data sets on a library tape and have the computer center copy them into a disk file for short periods of time when continual access is desired.

---

*When a tape is created or stored in the computer center, a number is assigned for purposes of cataloging and identification.

**Most data tapes will be created as labeled tapes with certain specifications (e.g. blocking factor, record size) which are contained on the tape. When the computer reads the FILE8 statement, it automatically searches for that label and reads the data - no operator intervention is required other than mounting the tape. In some cases, however, data tapes are unlabeled (no name and no specifications) and require operator intervention. Because the procedures involved with such tapes are as individual as the tapes, themselves, they will not be addressed in the manual. To deal with unlabeled data tapes, contact the Computer Center.

To enter raw data into a disk file, a data file can be made in the sequence mode and each line of data can be typed in from the terminal directly. If data is stored on a library tape, a request can be made to have it copied into a raw data disk file under a specified name. The following program illustrates this procedure:

```
BEGIN JOB__________________; CHARGE=_____________;
USER=______________________/________;
DECK  file name; DATA
|
| data records
|
END JOB
```

The result of this program will be a raw data file named under the specified name on the DECK command. Only the data records will reside as they appeared in the original card deck.

The Status of the File for the Run

Each time an S.P.S.S. program is run, a temporary file is created. It is the user's option to leave the file temporary or retain the file permanently. This segment of the manual provides instruction for saving, accessing and manipulating S.P.S.S. system files.

1. Temporary files:

   If the user chooses not to save the file after a run, no file equate statement is necessary (other than the appropriate raw data file statement previously mentioned). In this case, the information generated by the run will be lost.

2. Creating a Permanent S.P.S.S. System File:

   When a 'SAVE FILE' command appears in the program, this corresponding file equate statement must be present:

```
FILE FILE4(KIND= (medium),TITLE=__________________, FLEXIBLE, PROTECTION=SAVE);
```

The "medium" upon which the file will be saved is most often (and most conveniently) 'DISK' although a file can be saved on 'TAPE' (If a tape is chosen, the user must notify the computer center to arrange that a tape is mounted and readied for the writing of the permanent file.) The 'PROTECTION= SAVE' specifies that the file is to be saved.

The following provides an example program set-up:

```
?BEGIN JOB_____________________;
RUN SPSS;
FILE FILE4(KIND=DISK,TITLE=SAVEFILE,FLEXIBLE,PROTECTION=SAVE);
FILE FILE8(KIND=TAPE,TITLE=DATATAPE);
DATA IOCR
FILE NAME      SAVEFILE
I
INPUT MEDIUM   TAPE
I
READ INPUT DATA
I
SAVE FILE
FINISH
?END JOB
```

This program will read data from a tape and save the S.P.S.S. file on disk permanently under the name SAVEFILE.

3. Accessing a Permanent S.P.S.S. System File

When a 'GET FILE' command appears in a program, the corresponding file equate statement is:

```
FILE FILE3(KIND= (medium), TITLE=____________________);
```

The 'medium' can be either 'DISK' or 'TAPE' (the medium upon which the file was saved). The following presents an example program:

```
?BEGIN JOB______________________;
RUN SPSS;
FILE FILE3(KIND= DISK, TITLE= SAVEFILE);
DATA IOCR
GET FILE       SAVEFILE
I
FINISH
?END JOB
```

To access a file and save another version of the file, include both file equate statements. The following program will get a file, modify it and save the modified file (while preserving the original permanent file):

```
?BEGIN JOB_____________________________'
RUN SPSS:
FILE FILE3(KIND=DISK,TITLE=SAVEFILE);
FILE FILE4(KIND=DISK,TITLE= NEWFILE,FLEXIBLE,PROTECTION=SAVE);
DATA IOCR
FILE NAME      NEWFILE
GET FILE       SAVEFILE
I
I
I (S.P.S.S. operations including computes, records, etc.)
SAVE FILE
FINISH
?END JOB
```

4. Manipulating Permanent Files (Merging all or parts of various files):

To create one unified S.P.S.S. system file of all or a subset of variables from two to five existing permanent files, use the following file equate statements:

```
FILE FILE21(KIND= (medium), TITLE=_________________);
FILE FILE22(KIND= (medium), TITLE=_________________);
  .    .    .     .      .
  .    .    .     .      .      .
  .    .    .     .      .      .
                                .
FILE FILE25(KIND= (medium), TITLE=_________________);
```

Remember, the user must follow the rules associated with the 'MERGE FILE' command (e.g. each file must have the same number of cases in the same order; in addition, no more than 500 variables may be referenced). The order in which the files appear on the file equate statements must be the same order on the 'MERGE FILES' command. The following illustrates a program that merges two S.P.S.S. system files:

```
?BEGIN JOB____________________;
RUN SPSS:
FILE FILE21(KIND=DISK, TITLE=SAVEFILE);
FILE FILE22(KIND=DISK, TITLE=NEWFILE);
DATA IOCT
MERGE FILES      FILE=SAVEFILE  VARIABLES=ALL/
                 FILE=NEWFILE   VARIABLES=NEWVAR1,NEWVAR4
                 TO NEWVAR18/

I

FINISH
?END JOB
```

To save the unified (merged file), include the appropriate file statement (FILE4).

```
?BEGIN JOB ________________________;
RUN SPSS;
FILE FILE21(KIND=DISK,TITLE=SAVEFILE);
FILE FILE22(KIND=DISK,TITLE=NEWFILE);
FILE FILE4(KIND=DISK,TITLE=MERGFILE,FLEXIBLE,PROTECTION=SAVE);
DATA IOCR
FILE NAME       MERGFILE
MERGE FILES     FILE=SAVEFILE  VARIABLES=ALL/
                FILE=NEWFILE   VARIABLES=ALL/

I

SAVE FILE
FINISH
?END JOB
```

For cases in which more than 500 variables are being manipulated, archive files are available in S.P.S.S. The command in the above example would change from 'SAVE FILE' to 'SAVE ARCHIVE' although the file equate statements remain the same. The file statements used for 'MERGE FILES' are also used for 'GET ARCHIVE.'

NOTE: On Accessing and Manipulating Files that reside in Someone Else's Library.

Any job file or S.P.S.S. system file is created under a private usercode that is, in essence, part of the file name. It is possible (through CANDE) to alter the status of the private file by changing the security to allow anyone to read and/or write on the file if the entire name is known (for the procedure to change security, see Appendix 3).

The user may start, save, get and/or manipulate anyone else's public files using the same statements previously presented with one additional piece of information: the usercode under which the job file or system file was created.* The following illustrates the appropriate syntax:

```
GET (usercode) file name
START (usercode) file name
```

for job files

The above example gets someone else's system file and creates the user's own system file from it.

---

*The file name always begins with the other person's usercode in parenthesis.

If a file is not private, another person can copy, read and/or alter the file by knowing the name of the file and usercode under which it was created.

The Method of Handling Output

Output from an S.P.S.S. run can be sent to several "places" depending upon the user's preference and the nature of the output. An S.P.S.S. run can generate printed output and/or raw data output.

1. Printed Output

The default for printed output is the printer. If the appropriate file equate statement is not included in the program ro override this default, the output will be sent to the high speed line printer.

It is possible, however, to have the printed output sent to the remote station (terminal) from which the user is running the program. The appropriate file equate statement is:

```
FILE FILE6(KIND=REMOTE,TITLE=__________________,MAXRECSIZE=22);
```

This statement must appear after 'RUN SPSS' and before 'DATA IOCR.' The title of the terminal is immediately announced upon logging into the system. The following statement will appear:

```
B6800:  session #  CANDE______________; YOU ARE TTYLEASE1(22);
```

The part of the name to be included in the file statement is the first part only. The number in parenthesis is the logical station number and is not considered to be part of the title. 'MAXRECSIZE' is the command that informs the system as to the length of each statement (thereby making sure the system has allotted enough space for the lines of output. 22 is the correct number.)

When the program is started, this message will appear at the terminal:

```
#OUTPUT FILE FILE6 OPEN:  USER= "________________," PROG = *SPSS ON ACADEMIC
#RESPOND "OK" or "DENY"
```

If the user responds 'OK', the output will begin to appear. If the user does not want the output to appear, 'DENY' should be typed (when the # appears, the user may proceed with other tasks).

NOTE: If the user has changed terminals after the FILE FILE6 has been typed in, the title of the present terminal must be inserted in the file statement.

Another option for handling printed output is available. Similar to having printed output sent to a remote terminal, the user may direct the output to a disk file. The advantages of this option are:

a. The user may view selected sections of the output from the terminal without having to wait for the entire results to be printed on the terminal.
b. The output may be removed immediately if it is a mistake or not valuable without having to wait for it to be completely printed.
c. The user is being economical by not wasting computer paper on possibly useless output (or output that will be scanned then discarded).
d. The output is automatically available in the disk file for efficient user access.

The syntax for this option is:

```
FILE FILE6(KIND=DISK,TITLE=__________________________,FLEXIBLE, PROTECTION=SAVE,
          MAXRECSIZE=22);
```

After the S.P.S.S. run has been completed, the user may get the disk file with the specified title through the appropriate CANDE command and list all or parts of the output or perform any operation allowed on a disk file.

2. Raw Data Output

If an S.P.S.S. run is designed to produce raw data as output (e.g. WRITE CASES), a FILE9 statement must be used. The user may store the created raw data file either on a tape or in a disk file. The general format is:

```
FILE FILE9(KIND= (medium), TITLE=_______________, MAXRECSIZE=80,
          BLOCKSIZE=7200,UNITS=CHARACTERS, FLEXIBLE);
```

This statement will store the created raw data file as card images (maximum of 80 characters per record) on the medium of choice. The blocking factor is included to make reading the new data more efficient for the system but is optional.

NOTE: If a tape is chosen, the computer center must be notified to reserve a tape and ready it for the data-generating run.

In addition to the two above modes of storing a raw data file, the newly written data can be sent to the high speed line printer. In this way, the user has a paper copy of the new data to veiw and check before it is stored permanently on either a tape or disk. The file statement would look like:

```
FILE FILE9(KIND=PRINTER,TITLE=__________,MAXRECSIZE=80,
          UNITS=CHARACTERS,FLEXIBLE);
```

NOTE: When the user decides to store the data on tape or disk, the program must be re-run changing the FILE9 statement to the above format.

It is possible to create several (up to seven) raw data files in one S.P.S.S. run by including more file equate statements. With each request it is advisable to include a RAW OUTPUT UNIT statement to avoid confusion as to where the file is to be written.

```
FILE FILE9(KIND=_________,TITLE=_____________, . . .,FLEXIBLE);
FILE FILE15(KIND=________,TITLE=_____________, . . .,FLEXIBLE);
FILE FILE16(KIND=________,TITLE=_____________, . . .,FLEXIBLE);
 .     .     .      .       .        .                  .
 .     .     .      .       .        .                  .
                    .                .
FILE FILE20(KIND=________,TITLE=_____________, . . .,FLEXIBLE);
```

Running S.P.S.S. on Cards

There are basically two ways in which to create an S.P.S.S. run: by creating a job file from the terminal that houses the S.P.S.S. program or by keypunching the program on cards. Thus far, this manual has been written for the terminal user who makes, saves and runs S.P.S.S. from the terminal using disk space. The "old-fashioned" but still-used method of submitting a program "Batch" will be presented in this section for those who prefer using cards.

Very few differences exist between a job file and a card deck in the basic program set-up. In fact, only one major control statement placement differs: the "USER" statement. Because this statement is required when logging onto the system by terminal, it need not be repeated in the job file - the user has already been identified. When using cards, however, no such prior mechanism for identification exists. It is necessary, therefore, to include the "USER" statement in the program. The placement is as follows:

```
!BEGIN JOB______________________________; CHARGE=____________________;
 USER=________________/__________________;
 RUN SPSS;

   file equate statements (if required)

 DATA IOCR

   SPSS program

!END JOB
```

The following is a list of things to remember when submitting a Batch job:

1. The first column of the first and last card have a multipunch in the first column (at least 3 keys punched in one column) to serve as an illegal character (a '?' won't work).
2. The first card should be keypunched on a color-coded card (pink for faculty) which serves to mark the deck for the operators in the computer center. These pink cards are available at the computer center (5th floor of Kehoe).
3. To run a card deck, the user must go to the computer center and is required to fill out a "Job Request Form" (also available at the computer center). In this way, usage of the Batch system is documented.
4. The operators will run the program and the user will be notified when the output is ready. The user is responsible for collecting the deck and output.
5. Also available in the computer lab (basement of FOB II), is a RJE set-up where the user may run his/her own programs and get immediate output. If this facility is used the first card need not be color-coded and no job request form is required.

APPENDIX 1

Location of Terminals

There are two types of terminals available: hard-copy (paper) terminals and CRT (Cathode ray or screen) terminals. The hard-copy terminals are given faculty/staff priority while students are given priority on the CRT terminals.

A total of eight hard-copy terminals are interspersed throughout the campus:

3 in FOB II #143 (within the computer science department)
1 in Hawkins Hall #029 (basement)
2 in Hudson Hall #212
2 in Beaumont Hall #205-207

The terminals in FOB II and Hawkins Hall are available on a "sign-up" basis* while the remaining terminals are operated on a "first-come, first serve" basis. The terminals are available during normal office hours.

Located in the basement of FOB II are 20 CRT terminals available on a first-come, first-serve basis with a half-hour time limit and signing in procedure.

---

*To sign up for these terminals, either go to the secretary's office and sign the time sheet or call the secretaries who can schedule a time for you.

FOB II: Call 564-2788 (Department of Computer Science)
Hawkins Hall: 564-2122 (Department of Education)

APPENDIX 2

Basic Operating Instructions for the Hard-Copy and CRT Terminals

Hard-Copy Terminal Operation

1. Turn on the terminal by flipping up the switch on the lower left side of the back of the terminal. (The carriage will travel the width of the terminal once then "beep" when it has returned to starting position.)
2. Type in HELLO and press the "return" key. The computer will respond with:

```
#B6800:  session number CANDE 32.360; YOU ARE terminal name (XX)
#ENTER USERCODE PLEASE
```

3. Type in your usercode and press the "return" key. The computer will respond with:

```
#ENTER PASSWORD PLEASE
XXXXXXXXXXXXXXXXXXXXXX
```

4. Type in your password (over the crossed-out line which makes the password unintelligible) and press "return."

   The computer will respond with a series of messages pertaining the the last "log-off" date and time and the user's status then this message:

```
#ENTER CHARGECODE PLEASE
```

5. Type in your 5 digit departmental charge number and press "return."

   The computer will respond with messages about the most recent "news" and:

```
#SESSION session# time date
```

6. You are now logged onto the system and may perform any operations available in CANDE and/or WFL.
7. Log off the system by typing BYE and press "return."

```
#END SESSION session # ET=____  PT=____  IO=____
#USER=______________  CHARGE=________  time date
```

Special Features on the Hard-Copy Terminals

The hard-copy terminals are equipped with several special features to aid in program construction and editing.

a. The lower case keys are similar to that of a typewriter with special characters available in the upper case (when the "SHIFT" key is held). On these terminals, however, the special characters that will register are not the ones printed above the lower case character (as in a typewriter) but are illustrated on the front of the key in small print.

b. A "BREAK" key is available to "freeze" or stop the output from being printed. Just press the key once to stop the terminal then, when ready to begin, press "RETURN."

c. A "BACKSPACE" key allows the user to back up the carriage and re-type a mistake. Backspacing has the effect of "erasing" what was there and replacing it with new characters (not physically, but within the system).

d. To cancel out an entire line and start anew, press the "DEL" key.

CRT Terminal Operation

1. Clear the screen by pressing the 'SHIFT' then the 'CLEAR' keys.
2. Type HELLO and press the 'SEND LINE' key (upper right side of the keyboard).

Steps 3. through 7. are the same as for operating the hard-copy terminals with the exception that the 'SEND LINE' key is pressed NOT the 'return' key.

8. clear the screen as in #1.

Special instructions for the CRT terminals

Unlike the hard-copy terminals that will continue printing the output until finished, only a limited amount of information can appear on the CRT screen at one time. Once the screen is "full" output will stop even if there is more to come. To handle this feature of the terminal, remember the following:

a. To get more output after the screen is full, press the 'SEND LINE' key once. The terminal will then display the next section of information until either it is finished (a # appears) or it stops again (meaning the screen is full and "SEND LINE" must be pressed again).

b. If you type in a command when there is still more information to come from the previous command, this message will appear:

```
#QUEUED, SEND NULL INPUT FOR NEXT PAGE
```

Press the "SEND LINE" key thereby freeing the screen to display the remaining information and wait for the system to respond to the command typed in prematurely.

c. If the terminal "beeps," it is trying to display a message from the system. To receive the message, press 'SEND LINE.' To preserve the page being worked on so the message does not interrupt, the message can be displayed on the alternative page on the terminal. Use this sequence:

> Press the "Shift" key and the "new page" key simultaneously. Now press 'SEND LINE' to receive the message. To obtain the original page, press the "shift" and "new page" keys once again. In effect, the 2 pages can be "flipped" each time these two keys are pressed together.

d. To "freeze" output being continuously displayed, push the "shift," key and while continuously holding the "shift," press the "break" key (upper right of the key board). To "unfreeze" the screen, release both keys.

More information on editing with the CRT terminals is documented in 2 references available in the Department of Computer Science (See Appendix 4)

APPENDIX 3

Basic CANDE Commands

The following will present the most commonly used CANDE commands for editing and manipulating the S.P.S.S. program stored in a job file. For sources that provide a complete presentation of every CANDE and WFL command and all of their options, see the reference list.

Editing Commands

1. To see what the program looks like, type (after it is in the workspace using the GET command):

`LIST`	will list the entire contents of the job.
`LIST 100`	will list only that line.
`LIST 100-700`	will list lines 100 through 700.
`LIST 100-3000 60-72`	will list only the contents of columns 60 through 72 for lines 100 through 300.
`LIST: A`	will list only the altered lines in the entire job. Deleted lines are preceded by (-). Fixed lines " " " (F). New lines " " " (I). Lines that replaced other lines are preceded by a (R).
`LIST: CHANGES`	Same as A, but only the sequence numbers of deleted lines are listed (not their text.)
`LIST END`	will print the last line of the program.

2. To delete a line (or lines) from the program, two methods are available:
   a. For single line deletion, just type the line number then send the line (PRESS THE SEND LINE or RETURN key) which, in effect, sends an empty line or erases it.

Notice, the RUN SPSS command is missing. To insert it in the correct place, assign a line number that would place it in sequential order. Type:

```
150RUN SPSS;
```

In addition, your file statement is wrong and you choose to re-do the line. Type:

```
200FILE FILE3(KIND=TAPE,TITLE=FILE);
```

and list the lines:

```
LIST 100-300
```

```
100 ?BEGIN JOB FIRSTJOB;
150 RUN SPSS;
200 FILE FILE3(KIND=TAPE,TITLE=FILE);
300 DATA IOCR
```

5. To locate a particular string of characters (e.g. variable name, a mathematical procedure, a special character), type:

```
FIND /string/
```

The system will print the line number(s) on which the string occurs with a * denoting any line in which it appears more than once.

```
FINE /string/ :T
```

will also print out the actual line(s).

Example: You have these lines stored in a job file:

```
11900 RECODE     VARA,VARB (0 THRU 5 = 1)
12000 COMPUTE    VARSUM = 0
12100 IF         (VARA GT VARB) VARSUM = VARSUM + 1
12200 IF         (VARA EQ VARB) VARSUM = VARSUM + 2
```

b. For single and multiple line deletions, type:

```
DELETE #
```

will delete only that line.

or

```
DELETE # - #
```

will delete the first line through the last line numbered.

```
DELETE ALL
```

will delete every line in the job file.

3. To make a correction on a line in the program, type:

```
FIX line # /mistake/correction
```

Example: You have typed this first statement:

```
100 ?BEGIM JOB FIRSTJOB;
```

To correct the error, type

```
FIX 100 /BEGIM/BEGIN
```

Which will result in this corrected statement when line 100 is listed:

```
LIST 100
100 ?BEGIN JOB FIRSTJOB:
```

NOTE: The '/' is a delimiter. Any unique character can be used as a delimiter with the exception of ';' ',' or 'a'

4. To accomplish single line replacements or additions, type a line number then the line. To replace a line, type the same line number; to add a line, type a line number that would fall in numeric order to where the statement should be.

Example: You have this series of statements:

```
100 ?BEGIN JOB FIRSTJOB;
200 FILE FILE3(KIND=DISK,TITLE=DATA);
300 DATA IOCR
```

To have the system locate VARSUM for you without having to list the entire job file, get the file and type:

```
FIND /VARSUM/
```

The system will respond with:

```
#FILE   (USERCODE) file name on ACADEMIC
  12000,*12100,*12200
#
```

6. To replace a particular string of characters with another set of characters throughout all or part of the program, type:

```
REPLACE /old string/ /new string/
```

```
REPLACE /old string/ /new string/:  S
```

will replace the old with the new and print the numbers of the lines that were altered.

```
REPLACE /old string/ /new string/:  T
```

will replace and print the actual lines that were replaced.

```
REPLACE /old string/ /new string/# - #
```

will replace only in the line numbers specified.

7. To move lines in your program from one position to another, type:

```
MOVE line # - line # TO line # + inc
```

Examples:

```
MOVE 500 to 18000
```

will move line 500 to the position of 18000 and assign it that sequence number.

```
MOVE 300-600 to 900
```

will move lines 300,400,500,600 to 900,1000,1100,1200.

```
MOVE 300-600 to 900 + 20
```

will move 300,400,500,600 to 900,920,940,960.

8. To copy the lines of all or part of a program, type:

```
INSERT line# - line# AT line# + inc
```

Examples:

```
INSERT 100-500 AT 10000
```

will copy lines 100 through 500 and assign them line numbers 10000,10100,10200,10300,10400.

```
INSERT 100-500 AT 1000 + 1
```

will copy the lines and place them at 1000, 10001,10002,10003,10004.

```
INSERT 100-500 OTHERFILE AT 100
```

will copy lines 100-500 of another job file in your library and place them at 100-500 in the present work file.

```
INSERT 100-500 AT END
```

will copy lines 100 through 500 and place them at the end of the job file.

File Manipulation Facilities

1. Changing the security on a file in your library is accomplished by typing:

```
SECURITY file name keyword
```

Examples of Options:

```
SECURITY file name PUBLIC SECURED
```

will make a file public for execution only.

```
SECURITY file name PUBLIC IN
```

will make a file public for execution and reading only.

```
SECURITY file name PUBLIC OUT
```

will make file public for writing only.

```
SECURITY file name PUBLIC IO
```

will make file public for writing, reading and execution.

```
SECURITY file name PRIVATE
```

will prevent any other user to access the file.

The security of any file is originally PRIVATE. However, if the status is changed to PUBLIC, it can be changed back at any time. NOTE: a public file will not allow another user to: change its title, change its security status or file attributes, or remove the file.

2. When working on a terminal, it is often convenient to have an entire listing of the job file on paper. To get a printed copy (or several copies) of any library file, type:

```
WRITE file name
```

```
WRITE
```

will print every line of the file on the work space on the line printer.

```
WRITE 100-600, 1000-END
```

will print only the specified lines of the work file.

```
WRITE file name 3000-END @ 1-60
```

will print only columns 1-60 of the specified lines of the specified file.

```
WRITE:  COPIES = 3
```

will print 3 copies of the entire work file.

3. Often-times, you may want to make experimental or temporary changes in a program but want to retain the original as well. CANDE allows you to GET a job file or SAVE a job file under another name thereby creating 2 copies with which you can work.

   To get a job file under another name, type:

```
GET original job name AS new job name
```

   To save an edited file under another name, type:

```
SAVE original job name AS new job name
```

NOTE: Whenever a line is sent to the system, CANDE inserts a blank between the sequence number and first character in the line. Each time a line is present, the blank is read as part of the line and another blank is inserted (creating 2 spaces before each line or starting the statement in column 2). Since S.P.S.S. commands must begin in column one, it is very important to remove the first blank whenever a line is re-sent through CANDE.

S.P.S.S. USER'S MANUAL #4*

Quick File Equate Statement Reference

This manual is designed to provide the S.P.S.S. user with a quick reference for the B6800 file equate statements necessary for a variety of user needs. This reference presumes advanced knowledge of S.P.S.S. and the B6800 computing system and is intended as a "refresher" for information previously documented. Complete documentation of S.P.S.S. in general and S.P.S.S. on the Buroughs B6800 is presented in Manuals #1, #2 and #3 in this series.

This manual presents a file equate statement reference chart including all of the alternatives a user has for the form in which the data exist, the status of the generated file and the destination for the output. Following the chart will be several examples of situations outlined on the chart to illustrate how to use this reference. The situations will be numbered with the same numbering scheme from the chart and only selected situations will be illustrated.

---

*Prepared by:

Lorna L. Earl, M.S.
Spring 1982
SUNY at Plattsburgh
Office of Academic Computing

FILE EQUATE STATEMENT REFERENCE CHART

	Form of Data	Status of generated file	Output Destination	File statements to include
1	Raw Data on Cards (w/in program)	temporary	line printer	NONE
2	"	"	remote	FILE FILE6
3	"	"	disk	FILE FILE6
4	"	"	written cases	FILE FILE9, 15-20
5	"	permanent (SAVE FILE)	line printer	FILE FILE4
6	"	"	remote	FILE FILE6,FILE FILE4
7	"	"	disk or tape	FILE FILE6,FILE FILE4
8	"	"	written cases	FILE FILE4,FILE FILE9, 15-20
9	Raw Data on TAPE or DISK	--same statements as 1-8 but add this file statement to each		FILE FILE8
10	System file (GET FILE)	temporary	line printer	FILE FILE3
11	"	"	remote	FILE FILE3,FILE FILE6
12	"	"	disk	FILE FILE3,FILE FILE6
13	"	"	written cases	FILE FILE3,FILE FILE9, 15-20
14	"	permanent	line printer	FILE FILE3, FILE FILE4
15	"	"	remote	FILE FILE3,FILE FILE4 FILE FILE6
16	"	"	disk or tape	FILE FILE3,FILE FILE4 FILE FILE6
17	"	"	written cases	FILE FILE3,FILE FILE4 FILE FILE9, 15-20
18	Several System files (MERGE FILES)	temporary	line printer	FILE FILE21-25
19	"	"	remote	FILE FILE 21-25 FILE FILE6
20	"	"	disk	FILE FILE21-25, FILE FILE6
21	Several System files (MERGE FILES)	temporary	written cases	FILE FILE21-25 FILE FILE9, 15-20
22	"	permanent	line printer	FILE FILE21-25,FILE FILE4
23	"	"	remote	FILE FILE21-25,FILE FILE4 FILE FILE6
24	"	"	disk or tape	FILE FILE21-25,FILE FILE4 FILE FILE6
25	"	"	written cases	FILE FILE21-25,FILE FILE4 FILE FILE9, 15-20

Situation 1: Raw data records included within the program, permanent file not desired, output to high speed line printer.

```
?BEGIN JOB __________;
RUN SPSS;
DATA IOCR
|
|    S.P.S.S. program
|
INPUT MEDIUM        CARD
|
|
READ INPUT DATA
|
|  data records
|
|
|
FINISH
?END JOB
```

Situation 6: Raw data records included within the program, permanent file desired, output to a remote station (terminal).

```
?BEGIN JOB _________________;
RUN SPSS;
FILE FILE4(KIND= (medium),TITLE=__________,FLEXIBLE, PROTECTION=SAVE);
FILE FILE6(KIND=REMOTE,TITLE=_____________,MAXRECSIZE=22);
DATA IOCR


FILE NAME         _________
|
|
INPUT MEDIUM    CARD
|
|
READ INPUT DATA
|
|  data records
|
|
SAVE FILE
FINISH
?END JOB
```

Situation 9: Raw data records stored on a medium (TAPE or DISK) other than within the program, permanent file desired, output to a disk file. (Same as situation #7):

```
?BEGIN JOB_____________;
RUN SPSS;
FILE FILE8(KIND= (medium), TITLE=________________);
FILE FILE4(KIND= (medium), TITLE=________________, FLEXIBLE, PROTECTION=SAVE);
FILE FILE6(KIND=DISK, TITLE=_____________________, FLEXIBLE, PROTECTION=SAVE,
           MAXRECSIZE=22);
DATA IOCR
FILE NAME           __________
I
INPUT MEDIUM    (medium)
I
READ INPUT DATA
|
|
|
|
SAVE FILE
FINISH
?END JOB
```

Situation 10: Raw data as a system (SAVE) file, permanent file not desired, output to high speed line printer.

```
?BEGIN JOB________________;
RUN SPSS;
FILE FILE3(KIND= (medium), TITLE=_________);
DATA IOCR
GET FILE            _________
I
FINISH
?END JOB
```

Situation 13: Data as a system file, permanent file not desired, output in the form of 2 sets of written cases (generation of a new set of raw data records).

```
?BEGIN JOB________________;
RUN SPSS;
FILE FILE3(KIND= (medium), TITLE=__________________);
FILE FILE9(KIND= (medium), TITLE=__________________, MAXRECSIZE=80,
           BLOCKSIZE=7200,UNITS=CHARACTERS,FLEXIBLE);
FILE FILE15(KIND=(medium), TITLE=__________________, MAXRECSIZE=80,
           BLOCKSIZE=7200,UNITS=CHARACTERS,FLEXIBLE);
DATA IOCR
GET FILE          ____________
I
WRITE CASES       ______________________________________________
I
WRITE CASES       ______________________________________________
I
FINISH
?END JOB
```

Situation 15: Data as a system file, permanent file desired, output to a remote station.

```
?BEGIN JOB________________;
RUN SPSS;
FILE FILE3(KIND= (medium), TITLE=__________________);
FILE FILE4(KIND= (medium), TITLE=__________________,FLEXIBLE,PROTECTION=SAVE);
FILE FILE6(KIND=REMOTE,TITLE=________________,MAXRECSIZE=22);
DATA IOCR
FILE NAME        __________
GET FILE         __________
I
SAVE FILE
FINISH
?END JOB
```

Situation 18: Manipulation of several system files in one run (MERGE FILES), permanent file not desired, output to high speed line printer.

```
?BEGIN JOB________________;
RUN SPSS;
FILE FILE21(KIND= (medium), TITLE=________________);
FILE FILE22(Kind= (medium), TITLE=________________);
FILE FILE23(KIND= (medium), TITLE=________________);
DATA IOCR
MERGE FILES       FILE=_____________   VARIABLES=____________________/
|                 FILE=_____________   VARIABLES=____________________/
|                 FILE=_____________   VARIABLES=____________________/
|
|
FINISH
?END JOB
```

Situation 24: Manipulation of several system files, permanent file desired, output to a disk file.

```
?BEGIN JOB_______________;
RUN SPSS;
FILE FILE21(KIND= (medium), TITLE_________________);
FILE FILE22(KIND= (medium), TITLE=________________);
FILE FILE23(KIND= (medium), TITLE=________________);
FILE FILE4 (KIND= (medium), TITLE=________________,FLEXIBLE,PROTECTION=SAVE);
FILE FILE6 (KIND=DISK, TITLE=__________,FLEXIBLE,PROTECTION=SAVE, MAXRECSIZE=22);
DATA IOCR
FILE NAME         ______
MERGE FILES       FILE=____________   VARIABLES=________________________/
|                 FILE=____________   VARIABLES=________________________/
|                 FILE=____________   VARIABLES=________________________/
|
|
SAVE FILE
FINISH
?END JOB
```